Universidade do Minho
Escola de Engenharia

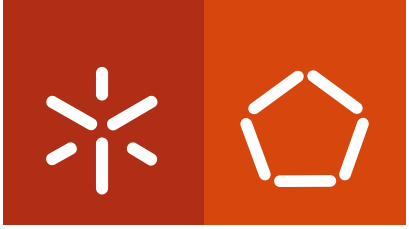
João Carlos Azevedo Salgado Guimarães

Design principles for controlling gene expression

Março de 2013

João Carlos Azevedo Salgado Guimarães **Design principles for controlling gene expression**

UMinho | 2013



Universidade do Minho
Escola de Engenharia

João Carlos Azevedo Salgado Guimarães

Design principles for controlling gene expression

Tese de Doutoramento em Informática

Trabalho realizado sob a orientação do
Doutor Miguel Rocha
e do
Doctor Adam Arkin

Março de 2013

Autor

João Carlos Azevedo Salgado Guimarães

Email: joaoguima@gmail.com

Teleforne: +351969886312

BI: 12626198

Título da tese

Design principles for controlling gene expression

Princípios de design para controlar a expressão genética

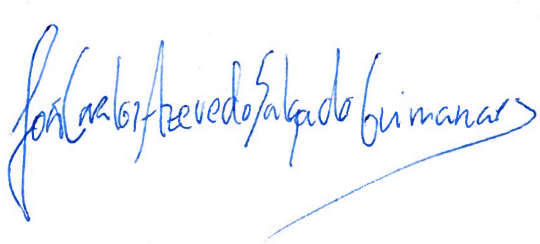
Orientadores

Doutor Miguel Rocha, Universidade do Minho

Doctor Adam Arkin, University of California, Berkeley

Ano de conclusão: 2013

Doutoramento em Informática



Universidade do Minho, Março de 2013

Acknowledgments

Completing this PhD has been a long journey that wouldn't have come to an end without the help of many people. I'm really thankful to all of you that helped me achieve this goal.

I'd like to thank Adam Arkin for receiving me at the University of California, Berkeley. The unmatchable experience to work in such an exciting and vibrant research environment has really broadened my horizons. I also thank Adam for being always so deeply engaged and teaching me how to improve my research.

I'm very grateful to Miguel Rocha for immediately accepting me as his PhD student and for letting me freely explore my own research interests. His expertise, patience and availability were essential to smooth my journey.

Most of all, I feel really privileged to have worked with Guillaume and Vivek, you were the best mentors I could have and I learned a lot from you.

The opportunity to collaborate with the BIOFAB was a remarkable research experience and I have to thank Vivek for inviting me to join the team. It was a true honor to be part of this incredible small group of smart people, who were able to produce high quality research in such a short period of time. It was also very fun to work with Quynh-Anh Mai and Colin Lam! Lastly, it was a very intense experience to work under the leadership of Drew Endy, whose dedication and vision are truly inspiring.

I also want to thank the entire Arkin lab, it was really fun to spend all this time surrounded by so many diverse and clever people. I'm particularly grateful to Gwyneth and Sara for being always available and helpful with administrative assistance, you made my life so much easier!

Quero ainda agradecer especialmente aos meus amigos da Bay area: Ana, Bernardo, Pat e Susana, aka 'Portuguese mafia', por todos os momentos de diversão e amizade. É sempre bom ter alguém que nos ajude a abstrair da bolha científica.

Aos meus pais e irmãs agradeço, do fundo do coração, o apoio incondicional, mesmo depois de lhes dizer que queria ser cientista, e por me recordarem o que de mais importante há na vida: família. Obrigado ainda pelas iguarias que volta e meia apareciam aqui à porta de casa.

Mais do que tudo, agradeço-te, Ana Luísa, pois, sem ti, nada do que consegui alcançar seria possível. Obrigado por teres despertado o meu interesse pela investigação com o teu entusiasmo e dedicação! Obrigado pela presença e apoio incondicional nos maus, que não foram poucos, mas, sobretudo, nos bons momentos. A teu lado, sinto que nada é impossível e que tudo vai correr bem.

This work was supported by an advance fellowship from Fundação para a Ciência e a Tecnologia through the program QREN - POPH (SFRH/BD/47819/2008). This work was also supported by the US National Science Foundation grant to the BIOFAB (EEC 0946510), the Synthetic Biology Engineering Research Center under the National Science Foundation grant 04-570/0540879, and the Joint BioEnergy Institute under the US Department of Energy contract DE-AC02-05CH11231.

Design principles for controlling gene expression

Abstract

Control of gene expression underlies the majority of cellular processes and, hence, it is of utmost importance to understand how living organisms tailor protein levels precisely at all times. In addition to comprehending how natural systems tune endogenous expression levels, it has recently become critical to develop genetic tools enabling reliable control of gene expression within synthetic circuits for biotechnology purposes. To this end, synthetic biologists seek parts (DNA segments) with diverse functional properties that once assembled together yield predictable behavior. Nonetheless, the design cycle of synthetic genetic circuits remains heavily dependent on multiple rounds of trial-and-error and manual tinkering.

One of the main hurdles faced by synthetic biology is the unpredictable behavior resulting from the reuse of genetic elements whose activities vary across changing contexts. Methods are lacking for researchers to affordably coordinate the quantification and analysis of part performance in different environments, as needed to identify, evaluate and improve problematic part types. We demonstrate how the combination of careful experimental designs and appropriate statistical frameworks can be used for quantifying the performance of genetic elements as they are reused in varying contexts. This methodology revealed design flaws of current gene expression platforms leading to unpredictable behavior. It further motivated the engineering of enhanced genetic elements that can reliably express sequence distinct genes across a 1,000-fold observed dynamic range and within twofold relative target expression windows with ~93% reliability.

Other than engineering efforts, a better understanding of how natural systems precisely control gene expression is equally important. However, living organisms

optimized by evolution are inherently complex and, commonly, difficult to understand. In this case, systems must be analyzed using integrative approaches that consider the multiple factors potentially affecting the observed phenotype.

To facilitate *in silico* analyses of these multi-factorial behaviors, we have developed an extendable software framework, D-Tailor, affording the automated inference of multiple relevant biological signals from plain genomic sequences. The software also implements a design module that allows researchers to generate artificial sequences exploring a wide range of parameters of interest so as to create more robust datasets to support the hypothesis being tested. We further demonstrated the validity of the above-mentioned integrative approach by evaluating more than 100 sequence features impacting translation efficiency across the *E. coli* genome, and also by exploring the determinants of specificity and functionality of the RNA-IN/OUT antisense RNA regulation system.

In summary, the work presented here shows how computational analysis frameworks can be efficiently combined with experimental approaches to get new insights into the design principles of natural and engineered genetic elements controlling gene expression. Such approaches will be essential for the engineering of more robust artificial systems and, ultimately, lead to the full understanding and modeling of natural biological systems.

Princípios de design para controlar a expressão genética

Resumo

O controlo da expressão genética sustenta a maioria dos processos celulares. Por conseguinte, é de extrema importância compreender como os organismos vivos produzem, nas concentrações adequadas, cada uma das suas proteínas. Atualmente, para além do interesse em compreender como se modelam os níveis de expressão endógenos, tornou-se crucial, para aplicações biotecnológicas, o desenvolvimento de sistemas que garantam um controlo fidedigno da expressão genética em circuitos sintéticos. Para este fim, a biologia sintética procura criar sequências de ADN (partes), com diversas funções, que exibam o comportamento esperado quando utilizadas em simultâneo. No entanto, o ciclo de design de circuitos genéticos sintéticos continua bastante condicionado pelo constante recurso a múltiplas iterações de tentativa-erro e ajuste manual.

Um dos principais obstáculos para a biologia sintética é a incapacidade de prever o comportamento resultante da reutilização de elementos genéticos cujas atividades variam em função do contexto. A escassez de métodos para quantificar e analisar a performance das partes nos diversos contextos limita a identificação, a avaliação e o aperfeiçoamento das partes problemáticas. Neste trabalho, combinamos design de experiências com métodos estatísticos apropriados para quantificar a variação da performance das partes à medida que estas são reutilizadas em diferentes contextos. A metodologia aplicada revelou falhas no design das plataformas de controlo da expressão genética atuais. Este fato motivou a conceção de novos elementos genéticos aperfeiçoados capazes de variar a expressão de genes com sequências distintas até três ordens de magnitude, e obter um determinado nível de expressão, numa janela até duas vezes o valor desejado, com uma probabilidade de 0.93.

Para além das soluções de engenharia, é, igualmente importante, a compreensão de como os sistemas naturais efetuam de forma precisa o controlo da expressão genética. Contudo, os organismos vivos otimizados pela evolução são inerentemente complexos e, comumente, de difícil compreensão. Nestes sistemas torna-se impreterível a realização de análises integradas que considerem os múltiplos fatores que afetam o fenótipo em estudo.

Para facilitar as análises *in silico* de comportamentos definidos por vários fatores, desenvolvemos uma plataforma de *software* extensível capaz de realizar a inferência automática de múltiplos sinais biológicos relevantes a partir de sequências genómicas. O *software* desenvolvido implementa ainda um módulo de design onde os investigadores podem gerar sequências artificiais que exploram uma grande variedade de parâmetros de interesse, criando assim conjuntos de dados mais robustos para a verificação da hipótese científica a ser testada. A análise integrada, acima descrita, foi usada para avaliar mais de 100 variáveis capazes de influenciar a eficiência da tradução na *E. coli*, e ainda para identificar os determinantes de especificidade e funcionalidade de um sistema de regulação da expressão genética por ARN.

Resumidamente, o trabalho apresentado demonstra como as análises computacionais podem ser combinadas com abordagens experimentais para elucidar os princípios de design de elementos genéticos, naturais e artificiais, que controlam a expressão genética. Tais abordagens serão essenciais para a engenharia de sistemas artificiais mais robustos e, futuramente, capazes de proporcionar a compreensão e a simulação de sistemas biológicos naturais.

Contents

Abreviations

A	adenine
AE	absolute error
Ala	alanine
ANOVA	analysis of variance
Arg	arginine
Asn	asparagine
Asp	aspartic acid
AU	arbitrary units
BCD	bicistronic design
BIC	Bayesian information criterion
bp	base pairs
C	cytosine
CAI	codon adaptation index
CDS	coding sequence
CRISPR	clustered regularly interspaced short palindromic repeats
CV	cross validation
Cys	cysteine
DB	database
df	degrees of freedom
DNA	deoxyribonucleic acid
EOU	expression operating unit
FCS	flow cytometry standard
FDR	false discovery rate
fMet	formylmethionine
G	guanine
GFP	green fluorescent protein
Gln	glutamine
Glu	glutamic acid
Gly	glycine
gspt	generated sequences per target
GTP	guanosine triphosphate
His	histidine

I	iodine
ICL	integrated completed likelihood
IF	initiation factor
Ile	isoleucine
IPTG	isopropyl-b-D-thiogalactopyranoside
Leu	leucine
Lys	lysine
MCD	monocistronic design
Met	methionine
MFE	minimum free energy
miRNA	micro RNA
MPL	modular promoter library
mRNA	messenger RNA
NN	neural networks
nts	nucleotides
OD_{600}	optical density at a wavelength of 600 nm
PA	protein abundance
PBS	phosphate buffered saline
PFM	position frequency matrix
Phe	phenylalanine
PLS	partial least squares
Pro	proline
PWM	position weight matrix
qPCR	quantitative polymerase chain reaction
RBS	ribosome binding site
RE	relative error
RF	random forest
RFP	red fluorescent protein
RFU	relative fluorescent unit
RNA	ribonucleic acid
RPL	randomized promoter library
rRNA	ribosomal RNA
SD	Shine-Dalgarno
SEM	standard error of the mean
Ser	serine
sRNA	small RNA
SVM	support vector machines
T	thymine
TE	translation efficiency
TF	transcription factor
Thr	threonine
TIE	translation initiation element
TIR	translation initiation region
tRNA	transfer RNA
Trp	tryptophan

Tyr	tyrosine
U	uracil
UML	unified model language
UTR	untranslated region
Val	valine
YUNR	pyrimidine-uracil-nucleotide-purine

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Single cells are tiny factories whose utmost goal is to grow and multiply. Every cell carries a blueprint of all the information necessary for a remarkable operation of all its constituents, which is ultimately encoded in its DNA (deoxyribonucleic acid). This persistent storage ensures that, during cell replication, all genetic information and corresponding traits are transmitted from parents to offspring. The genetic material encodes a myriad of rich information including genes, which control most of the aspects of the life cycle of an organism. Therefore, it is extremely important that RNA (ribonucleic acid) and protein expression levels are precisely tailored at all times.

Gene expression control systems face the grand challenge of ensuring the coordinated regulation of thousands of genes guided by the reciprocal exchange of chemical and physical stimuli from both internal (cellular) and external (environment) interactions. The flow of genetic information from DNA to proteins is attained by two steps: transcription and translation, and is commonly described as the central dogma of molecular biology. It is widely accepted that proteins have a major role in gene expression regulation by influencing transcription, translation and messenger RNA (mRNA) degradation. For example, the enzymatic protein RNA polymerase is responsible for the catalysis of RNA molecules from DNA segments (transcription).

Similarly, translation of mRNA into an amino acid sequence in proteins is performed by ribosomes, which are large complexes formed by proteins and RNA molecules. In addition, there is an emergent recognition of the relevance of RNA regulators (e.g., small RNAs), which have been shown to be pervasive in nature, to control gene expression (Brantl, 2002; Serganov and Patel, 2007; Wassarman, 2002).

Proteins are the main functional entities of the cell and their abundance results from the balance of multiple regulatory processes (Vogel and Marcotte, 2012). Dissecting the relative contribution of each process to the steady-state protein abundances has remained hampered by the difficulty to generate large-scale proteomics datasets. However, recent technological advances affording high-throughput quantification of the transcriptome and proteome provide valuable datasets for the examination of the multiple determinants of protein concentration.

The last 20 years have witnessed incredible progress in both our understanding of molecular biological systems and the technologies available for manipulating them. As a result, biology is quickly expanding from its historical tradition as a discovery-based science into an engineering science, where biological matter (e.g., RNA, proteins, regulatory circuits and cells) is treated as building material for the construction of custom biological functions. This transition has already brought biological solutions to advanced problems in chemical and pharmaceutical production (Keasling, 2010; Lee et al., 2009), therapeutics and diagnostics (Ruder et al., 2011), as well as agricultural and environmental engineering (Zhao and Chen, 2008). In fact, it has originated the new field of synthetic biology.

A fundamental principle of synthetic biology is the existence of biological parts, which are genetically encoded modular units with defined biological function (e.g., promoter, gene and transcription terminator) that can be used and reused in different contexts. Once collections of parts are available, the next level involves their predictable hierarchical assembly into composite circuits of greater and greater complexity (Endy, 2005). In fact, synthetic biology principles resemble those of other engineering disciplines, such as electric engineering where standard components (e.g., transistors and resistors) are assembled into integrated circuits to yield a specified design.

In theory, predictable assembly is feasible as long as parts are well characterized and behave consistently in a multitude of contexts. However, it has become clear

that parts are not always modular — i.e., the functions of many parts are not neatly encapsulated in the part itself but rather are significantly influenced by interactions with different contexts (Cardinale and Arkin, 2012). For example, a 5' untranslated region (5'UTR) can promote a strong translation initiation rate for a given gene, but accomplish only modest protein yield with a different coding sequence due to emergent inhibitory RNA secondary structures from the altered junction identity. In addition, even if parts do work as expected, it is not guaranteed that the resulting circuit will produce the expected behavior since many other uncontrollable factors (e.g., stochastic expression and host incompatibility) are at play (Kwok, 2010).

1.2 Aims

Gene expression control systems are of utmost importance to ensure the functionality of every genetic circuit by precisely producing the required levels of all molecular components. It is of crucial interest to understand what are the key determinants of expression level of endogenous genes, not only from a discovery perspective, but also for engineering purposes. The latter approach seeks the development of standard biological elements with encapsulated function that can be predictably assembled to create higher-complexity circuits.

This thesis focuses on the analysis of gene expression control systems from both a natural and engineering perspective. Therefore, the main goals to be sought in this work are the following:

1. To derive a quantitative framework wherein the performance and reliability of standard biological parts can be scrutinized in varying contexts. Such framework will lead to the identification of design flaws and motivate future iteration of parts design to either overcome or, at least, predict deviant behavior emerging from divergent context.
2. To improve biological parts design so as to create synthetic genetic elements controlling constitutive gene expression that are more reliable and predictable across multiple contexts.

3. To develop a framework to facilitate the automatic extraction of gene expression control elements from plain DNA sequences and enable *in silico* tailoring of DNA sequences with user-defined properties.
4. To interrogate the determinants of gene expression control systems in both endogenous genes and synthetic genetic circuits. The inference of novel and relevant factors will lead to a better understanding of natural systems and aid the rational design of the synthetic ones.

1.3 Thesis outline

This chapter presented a brief introduction to gene expression control systems and synthetic biology as well as the motivation and aims of this work.

Chapter 2 provides a general overview of the background knowledge that served as support to this thesis. It intends to cover some aspects of transcriptional, translational and RNA regulation control that are of extreme importance to cellular gene expression control. It also provides an introduction to the field of synthetic biology and many of the basic concepts that underlie this engineering discipline of biological systems.

Chapter 3 proposes a formal methodology for the characterization and analysis of artificial biological elements in varying contexts. It describes how experimental design can be combined with appropriate statistical frameworks to quantify parts performances and estimate their quality across changing contexts.

Chapter 4 shows a practical demonstration of how the knowledge inferred from quantitative assessment of parts quality (from Chapter 3) can guide future engineering efforts to create synthetic biological elements insulated from immediate genetic context. The result is a set of universal elements that enable reliable forward engineering of gene expression at the genome scale.

Chapter 5 introduces a software framework, D-Tailor, which allows the multi-dimensional examination of gene expression control elements encoded within DNA sequences. It aims to deliver a transparent and extendable architecture such that it can be easily adapted to new sequence-encoded features of interest. In addition, the software anticipates near-future needs resulting from readily available

DNA synthesis by implementing a flexible forward engineering platform for DNA sequence design.

Chapter 6 presents a thorough investigation of the sequence-encoded gene expression control elements in the endogenous genes of *E. coli*. It proposes the adoption of a statistical model to find a minimal set of determinants influencing protein abundances genome-wide. It further provides estimates of transcription and translation effects, as well as their relative contribution to protein concentrations.

Chapter 7 uses a similar analysis methodology to the one described in Chapter 6 but applied to artificial regulatory elements. A synthetic biology approach was used to generate multiple variants of a regulatory element from a natural system. Then, multiple determinants were analyzed to yield a sequence-activity relationship model highly predictive of synthetic elements performances. Further, the model was used to forward engineer new sets of synthetic regulatory elements with desired behavior.

Chapter 8 contains an overview of the work developed and conclusions derived from it. Lastly, it depicts general limitations of the work and suggests some directions for further improvement.

Chapter 2

Background

2.1 Central dogma: from genes to proteins

The central dogma of molecular biology describes the flow of genetic information from DNA to RNA, and then from RNA to proteins (Crick, 1970). Proteins mediate the majority of cellular processes that determine the physiological state of the cell and their abundance is a key measure of the corresponding activity. For that reason, protein levels have to be precisely tuned at all times to ensure that cellular needs are satisfied. Gene regulation is a dynamic process that is achieved through a myriad of different regulatory mechanisms such as transcription, translation, mRNA degradation, RNA regulation and posttranslational modifications, just to name a few. Here, we mainly focus on prokaryotic gene regulation events that occur at the level of transcription, translation and a specific type of RNA regulation.

2.1.1 Transcription

The enzyme RNA polymerase accomplishes the first step of gene expression. Its task is to catalyze a new RNA molecule using the DNA sequence as the template. The double stranded DNA unwinds, in regions close to the gene being transcribed, into two single stranded molecules, one of which is used to make an RNA copy. The result is an RNA molecule fully complementary to the template strand. The newly

synthesized RNA chain contains the nucleotide uracil instead of thymine, which can base-pair with adenine and guanine residues.

Different types of RNA molecules are catalyzed from genomic sequences: (i) mRNAs encode protein sequences; (ii) transfer RNAs (tRNAs) are responsible for transporting amino acids to ribosomes during translation; (iii) ribosomal RNAs (rRNAs), which together with ribosomal proteins, form the ribosome complex; (iv) other RNAs with catalytic and/or regulatory functions.

The process of transcription is devised in three consequential steps: initiation, elongation and termination. In *E. coli*, a transcriptional event is triggered by the interaction between a complex and a promoter region just upstream of the transcriptional start site (initiation). This complex is formed by the RNA polymerase and by another protein called the sigma factor, where the latter is the one responsible for the recognition of the promoter sequence. Then, transcription proceeds through the RNA-coding region (elongation) until it reaches a termination signal (Figure 2.1).

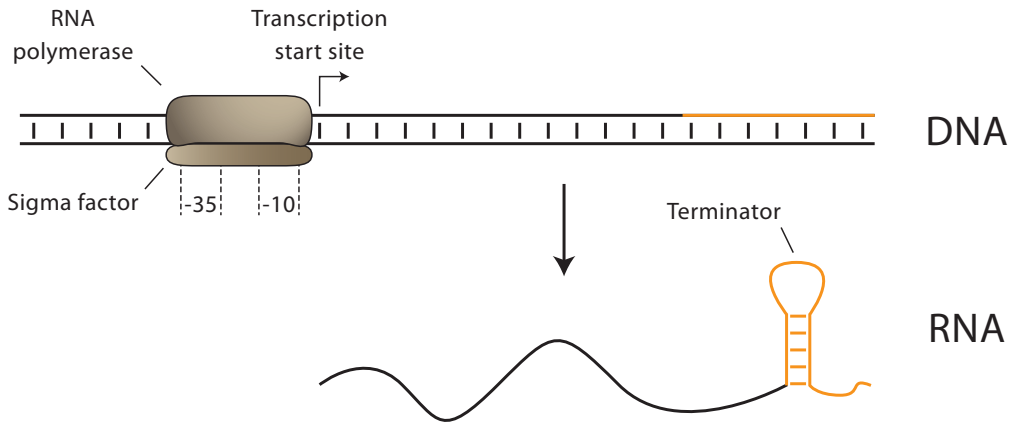


Figure 2.1 Transcriptional process
A complex formed by the RNA polymerase and sigma factor recognizes the promoter region, just upstream of the RNA-coding sequence, and proceeds for the elongation step until it finds a termination signal (orange).

Promoter regions typically comprise two critical DNA sequences for the sigma factor recognition. These sequences are generally found at positions -35 and -10 with respect to the transcriptional start site (+1). The consensus sequence (the sequence more frequently observed) for the -35 region is 5'-TTGACA-3' and for the -10 region is 5'-TATAAT-3'.

Promoters found in *E. coli* differ slightly in their sequence and, hence, their binding affinity to RNA polymerase may also vary. This can partially explain why endogenous genes are transcribed at widely divergent rates. Promoter sequences more similar to the consensus are stronger (higher transcription rates), whereas sequences deviating significantly from the consensus tend to be weak.

After the initiation step, RNA polymerase moves along the DNA double helix, untwisting it at the same time it synthesizes the new RNA molecule. Transcription termination in *E. coli* is known to occur via two distinct mechanisms: factor-dependent or factor-independent termination. Factor-dependent termination relies on the destabilization of transcription complexes by a regulatory protein, Rho, at Rho-dependent terminator sequences. A recent study showed that the Rho protein is responsible for ~20% of termination events in *E. coli* (Peters et al., 2009). Alternatively, factor-independent termination, which accounts for the remaining ~80% of transcription termination events in *E. coli,* occurs at defined sequence regions known as intrinsic terminators. These regions are characterized by GC-rich (high content of guanine and cytosine) sequences followed by a stretch of Ts (thymine) in the nontemplate DNA strand that, when transcribed, form a GC-rich hairpin structure followed by an Uracil-rich tract, which triggers the transcription termination signal (Figure 2.1).

2.1.2 Translation

Transcription of protein-coding genes is followed by translation of the respective mRNA, which consists in the conversion of transcript nucleotide information into the amino acid sequence of a polypeptide. The function of a newly synthesized protein is determined by both its amino acid sequence and three-dimensional shape.

There are twenty different amino acids that can be used to make proteins in living cells; their names along with the three-letter abbreviation can be found in Table 1. The mRNA nucleotide information that specifies a particular amino acid is called the genetic code. Since there are 20 amino acids and only four different nucleotides (A — adenine, C — cytosine, G — guanine, U — uracil), a three-letter code is necessary to generate all the required combinations. This code generates 64 (4^3) differ-

ent combinations, which leads to most amino acids being encoded by more than one codon (Table 2).

Table 1 The 20 natural amino acids
List of the 20 different amino acids along with their three-letter abbreviation and chemical type.

Amino acid	Abbreviation	Chemical type
Aspartic acid	Asp	Acid
Glutamic acid	Glu	Acid
Tryptophan	Trp	Neutral, nonpolar
Phenylalanine	Phe	Neutral, nonpolar
Glycine	Gly	Neutral, nonpolar
Alanine	Ala	Neutral, nonpolar
Valine	Val	Neutral, nonpolar
Isoleucine	Ile	Neutral, nonpolar
Leucine	Leu	Neutral, nonpolar
Methionine	Met	Neutral, nonpolar
Proline	Pro	Neutral, nonpolar
Lysine	Lys	Basic
Arginine	Arg	Basic
Histidine	His	Basic
Tyrosine	Tyr	Neutral, polar
Serine	Ser	Neutral, polar
Threonine	Thr	Neutral, polar
Asparagine	Asn	Neutral, polar
Glutamine	Gln	Neutral, polar
Cysteine	Cys	Neutral, polar

Ribosomes, complexes made of ribosomal proteins and rRNA, are responsible for translating the mRNA and synthesizing the respective protein. The mRNA is translated in the 5' → 3' direction and all proteins have start (codon AUG and more rarely GUG or UUG) and stop signals (codons UAA, UAG or UGA). During translation, the amino acids are brought to the ribosomes by charged tRNA molecules. The synthesized amino acid sequence results from the binding between each codon of the mRNA and the complementary anticodon in the tRNA that brings the corresponding amino acid attached. The interaction between anticodon and codon follows the wobble pairing rules. While the Watson-Crick base pairing rules posit that G can pair with C and A can pair with U or T (thymine), the wobble pairing rules indicate that, in RNA molecules, four other base parings can occur: G with U and I (iodine, formed after posttranslational modification) with A, U or C. Consequently, the same tRNA can recognize different mRNA codons. For example, the leucine

tRNA molecule with anticodon GAG can read two different codons (CUC and CUU) though it does so with different affinities.

Table 2 The genetic code
There are 64 different codons: 61 specify one of the 20 different amino acids, whereas the three others are terminating codons and do not specify any amino acid.

First letter	Second letter: U		Second letter: C		Second letter: A		Second letter: G		Third letter
U	UUU	Phe	UCU	Ser	UAU	Tyr	UGU	Cys	U
	UUC		UCC		UAC		UGC		C
	UUA	Leu	UCA		UAA	Stop	UGA	Stop	A
	UUG		UCG		UAG	Stop	UGG	Trp	G
C	CUU	Leu	CCU	Pro	CAU	His	CGU	Arg	U
	CUC		CCC		CAC		CGC		C
	CUA		CCA		CAA	Gln	CGA		A
	CUG		CCG		CAG		CGG		G
A	AUU	Ile	ACU	Thr	AAU	Asn	AGU	Ser	U
	AUC		ACC		AAC		AGC		C
	AUA		ACA		AAA	Lys	AGA	Arg	A
	AUG	Met	ACG		AAG		AGG		G
G	GUU	Val	GCU	Ala	GAU	Asp	GGU	Gly	U
	GUC		GCC		GAC		GGC		C
	GUA		GCA		GAA	Glu	GGA		A
	GUG		GCG		GAG		GGG		G

Similarly to transcription, three steps compose the translation process: initiation, elongation and termination. Initiation involves an mRNA molecule, a ribosome, an initiator tRNA, three initiation factors (IFs), guanosine triphosphate (GTP) and magnesium ions. The very first step of initiation is the binding of the 30S small ribosomal subunit (along with the IFs, GTP and magnesium ions) to a region just upstream of the start codon. This region, commonly known as the Shine-Dalgarno (SD) sequence (Shine and Dalgarno, 1975), binds to a complementary sequence at the 3' end of 16S rRNA in the small subunit of the ribosome allowing it to find the start codon and form the initiation complex (Figure 2.2). The next step is the binding of the initiator tRNA, which in prokaryotes specifies formylmethionine (fMet, a modified form of Met), to the start codon. After that, the 50S ribosomal subunit binds to the 30S complex giving origin to the final complex called the 70S initiation complex, which is then ready to start the elongation step.

Although it may look reasonable to assume that stronger binding between SD and 16S rRNA will result in higher translation rates (Boni, 2006), there is some ex-

perimental evidence that longer SD sequences do not necessarily increase protein yield (Vimberg et al., 2007). The distance between the SD sequence and the start codon — known as spacer — can also affect the initiation rate (Chen et al., 1994; Vellanoweth and Rabinowitz, 1992). Though it has been shown experimentally that *E. coli* has an optimum spacing of 7 to 9 nucleotides (Vellanoweth and Rabinowitz, 1992), the spacer size observed for the genes of this organism is highly variable ranging from 2 to 15 nucleotides and averaging around 7 (Shultzaberger et al., 2001). The identity of the start codon also influences the initiation efficiency, being AUG the preferred codon, followed by GUG and UUG (Vellanoweth and Rabinowitz, 1992). Lastly, inhibitory RNA secondary structures around the initiation region can hinder the formation of the initiation complex (de Smit and van Duin, 1990, 1994; Hall et al., 1982).

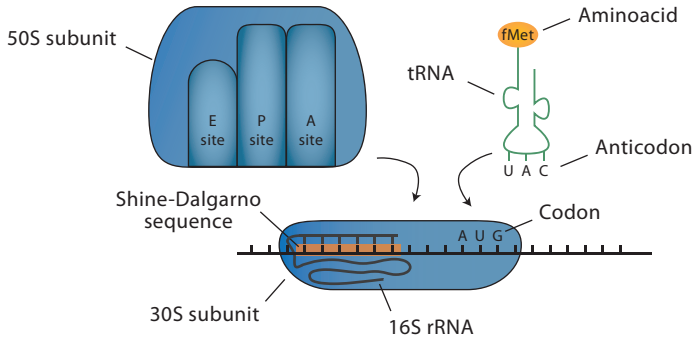


Figure 2.2 Translation initiation
The 16S rRNA in the 30S ribosomal subunit recognizes the Shine-Dalgarno (SD) sequence just upstream of the start codon before initiation starts. Then, the initiator tRNA binds to the start codon and, lastly, the 50S subunit binds to the 30S unit to form the final complex.

The elongation step follows the initiation one. At the start of elongation, the initiator tRNA is bound to the start codon in the peptidyl (P) site of the ribosome and the next codon to be read is in the aminoacyl (A) site. Next, the appropriate aminoacyl-tRNA binds to the codon in the A site, followed by the creation of a peptide bond between the two amino acids in positions P and A. At this time, the tRNA in the A site, now called the peptidyl-tRNA, will contain the chain of amino acids of the protein being synthesized. The elongation cycle contains one more step, translocation, where the ribosome moves to the next codon in the mRNA. In this step, the now uncharged tRNA moves from the P site to the exit (E) site, while the tRNA in position A (which contains the amino acid chain) binds to position P freeing the A

site, which is then ready to read the next codon. The entire elongation cycle is then repeated until the ribosome arrives to a stop codon. Once the ribosome moves away from the initiation site, free ribosomes can readily execute another initiation event. This way, many ribosomes can be attached to the same mRNA molecule, forming a complex called the polysome. A recent study, using a new ribosome-profiling method based on deep-sequencing to track ribosomes in RNA molecules with remarkable resolution, revealed that the first 30 to 40 codons of the gene had greater ribosome density (~3-fold) than the rest of the gene (Ingolia et al., 2009).

Lastly, termination of translation is signaled by one of the three stop codons (UAA, UAG or UGA), which are not read by any tRNA and, hence, do not code for any specific amino acid. Instead, specific proteins called release factors recognize the stop codons and initiate a set of termination events that culminate in the release of the newly synthesized protein and the ribosome subunits.

The redundancy of the genetic code — where the same amino acid can be translated by multiple codons (Table 2) — offers the opportunity to encode the exact same protein sequence using numerous different mRNA sequences. Such property may suggest an inconsequential use of the different synonymous codons (i.e., those encoding the same amino acid). However, the non-random utilization of such codons (codon bias) is pervasive in nature (Table 3). The natural selection theory (Bulmer, 1991) posits that such biases result from the adaptation to tRNA pools (Table 4) and are more noticeable in highly expressed genes because these are subject to a greater pressure for translation accuracy (Akashi, 1994) — production of error-free proteins — and efficiency (Ikemura, 1985). Although there is some evidence that codons are translated faster by more abundant tRNAs (Varenne et al., 1984), large-scale studies employing measurements of endogenous mRNA and protein levels have failed to show a correlation between translation efficiency and codon bias (Lu et al., 2007; Vogel et al., 2010). In fact, for endogenous genes, it is generally believed that initiation is the rate-limiting step of protein synthesis (Andersson and Kurland, 1990; Bulmer, 1991). This assertion is not only supported by the biophysics of translation (Bulmer, 1991), but also by the observation that ribosomes in a polysome are fairly spaced and by the lack of correlation between experimental data on the number of ribosomes bound to a particular mRNA and its translation efficiency (Belle et al., 2006; Beyer et al., 2004; Brockmann et al., 2007; Plotkin and

Kudla, 2011). Intuitively, it also looks more efficient to modulate the expression level of a gene by tuning the efficiency of a promoter and/or a ribosome-binding site rather than altering multiple codons of a gene to tweak its elongation rate.

Table 3 Codon bias in *E. coli*

Codon usage of the different codons in *E. coli* W3110. Ratio represents the usage of that codon relative to all codons coding that particular amino acid.

Codon	Amino acid	Ratio	Codon	Amino acid	Ratio	Codon	Amino acid	Ratio	Codon	Amino acid	Ratio
UUU	Phe	0.57	UCU	Ser	0.15	UAU	Tyr	0.57	UGU	Cys	0.44
UUC		0.43	UCC		0.15	UAC		0.43	UGC		0.56
UUA	Leu	0.13	UCA		0.12	UAA	Stop	-	UGA	Stop	-
UUG		0.13	UCG		0.15	UAG	Stop	-	UGG	Trp	1
CUU		0.10	CCU	Pro	0.16	CAU	His	0.57	CGU	Arg	0.38
CUC		0.10	CCC		0.12	CAC		0.43	CGC		0.40
CUA		0.04	CCA		0.19	CAA	Gln	0.35	CGA		0.06
CUG		0.50	CCG		0.53	CAG		0.65	CGG		0.10
AUU	Ile	0.51	ACU	Thr	0.17	AAU	Asn	0.45	AGU	Ser	0.15
AUC		0.42	ACC		0.44	AAC		0.55	AGC		0.28
AUA		0.07	ACA		0.13	AAA	Lys	0.77	AGA	Arg	0.04
AUG	Met	1	ACG		0.27	AAG		0.23	AGG		0.02
GUU	Val	0.26	GCU	Ala	0.16	GAU	Asp	0.63	GGU	Gly	0.34
GUC		0.22	GCC		0.27	GAC		0.37	GGC		0.41
GUA		0.15	GCA		0.21	GAA	Glu	0.69	GGA		0.11
GUG		0.37	GCG		0.36	GAG		0.31	GGG		0.15

Table 4 tRNAs abundances in *E. coli*
tRNA name, anticodon (corresponding codons) and abundance (adapted from (Dong et al., 1996)).

tRNA	Anticodon	Codon	Fraction of tRNA out of total tRNA (%)
Ala1B	UGC	GCU, GCA, GCG	5.04
Ala2	GGC	GCC	0.95
Arg2	ACG	CGU, CGC, CGA	7.37
Arg3	CCG	CGG	0.99
Arg4	UCU	AGA	1.34
Arg5	CCU	AGG	0.65
Asn	GUU	AAC, AAU	1.85
Asp1	GUC	GAC, GAU	3.72
Cys	GCA	UGC, UGU	2.46
Gln1	UUG	CAA	1.18
Gln2	CUG	CAG	1.36
Glu2	UUC	GAA, GAG	7.32
Gly1	CCC	GGG	3.31
Gly2	UCC	GGA, GGG	
Gly3	GCC	GGC, GGU	6.76
His	GUG	CAC, CAU	0.99
Ile1	GAU	AUC, AUU	5.39
Ile2	CAU	AUA	
Leu1	CAG	CUG	6.94
Leu2	GAG	CUC, CUU	1.46
Leu3	UAG	CUA, CUG	1.03
Leu4	CAA	UUG	2.97
Leu5	UAA	UUA, UUG	1.6
Lys	UUU	AAA, AAG	2.97
Met f1	CAU	AUG	1.88
Met f2	CAU	AUG	1.11
Met m	CAU	AUG	1.09
Phe	GAA	UUC, UUU	1.6
Pro1	CGG	CCG	1.38
Pro2	GGG	CCC, CCU	1.11
Pro3	UGG	CCA, CCU, CCG	0.9
Sec	UCA	UGA	0.34
Ser1	UGA	UCA, UCU, UCG	2.01
Ser2	CGA	UCG	0.53
Ser3	GCU	AGC, AGU	2.18
Ser5	GGA	UCC, UCU	1.18
Thr1	GGU	ACC, ACU	0.16
Thr2	CGU	ACG	0.84
Thr3	GGU	ACC, ACU	1.7
Thr4	UGU	ACA, ACU, ACG	1.42
Trp	CCA	UGG	1.46
Tyr1	GUA	UAC, UAU	1.19
Tyr2	GUA	UAC, UAU	1.95
Val1	UAG	GUA, GUG, GUU	5.96
Val2A	GAC	GUC, GUU	0.97
Val2B	GAC	GUC, GUU	0.98

2.1.3 RNA regulators

RNA molecules were historically regarded as only functioning as mRNAs or as being part of the translational apparatus (tRNAs and rRNAs). The first natural regulatory RNAs, discovered in 1981, were plasmid-encoded elements controlling the copy number of the *E. coli* plasmids ColE1 and R1 (Stougaard et al., 1981; Tomizawa et al., 1981). Now, we know that RNA regulators are extremely versatile molecules that can use a wide variety of mechanisms to control gene expression and play key roles in a plethora of important biological processes (Gottesman and Storz, 2011; Waters and Storz, 2009).

RNA regulators can target both proteins and mRNA molecules. In the former type of regulation, RNA elements can bind to and hamper the activity of the cognate proteins by resembling the structures of their common targets (Babitzke and Romeo, 2007). In the case where mRNA molecules are targeted, the regulatory element can be either part of the molecule it is regulating or an independent transcript acting on its target by base pairing. For example, riboswitches, an example of the first class, are RNA structures present in the 5'UTR of an mRNA that change their conformation in response to small molecules, thereby activating or repressing the expression of the corresponding gene (Mandal and Breaker, 2004). The second class of regulators is the most extensively studied, and an element in this category is commonly designated as a small RNA (sRNA). sRNAs can be split in two categories: those that are transcribed from the DNA strand opposite to their target (*cis*-encoded) and those that are transcribed from a different DNA region (*trans*-encoded) (Figure 2.3).

Cis-encoded (or antisense) RNAs, which are transcribed from the DNA strand opposite to their target, form a double helix with their targets exhibiting extensive complementarity. In contrast, *trans*-encoded sRNAs, which are transcribed from DNA regions different than the target, form duplexes exhibiting much more limited base pairing. sRNAs can bind to any region of the target mRNA molecule and either activate or repress gene expression through a variety of different mechanisms (Figure 2.3).

Regulation by sRNAs offers many advantages. First, the fact that a single sRNA can regulate many targets makes sRNAs particularly appealing as global

modulators of physiological responses such as heat shock or quorum sensing. Second, sRNAs are less costly and faster to produce than transcription factors (TFs) since they are relatively short and do not need to be translated. Third, sRNAs can exhibit rich regulatory functions, namely establish threshold linear responses, reduce low protein abundance fluctuations and implement complex hierarchical regulation between competing targets (Levine and Hwa, 2008; Levine et al., 2007).

Further, from an engineering standpoint, the apparent simplicity of RNA molecules base pairing rules and modularity of their structural components make them excellent substrates for design, and attractive contenders for a standard platform for gene expression regulation in synthetic biological applications Moreover, their ability to sense biomolecules and to act in *cis* or in *trans* allows the construction of complex regulatory networks.

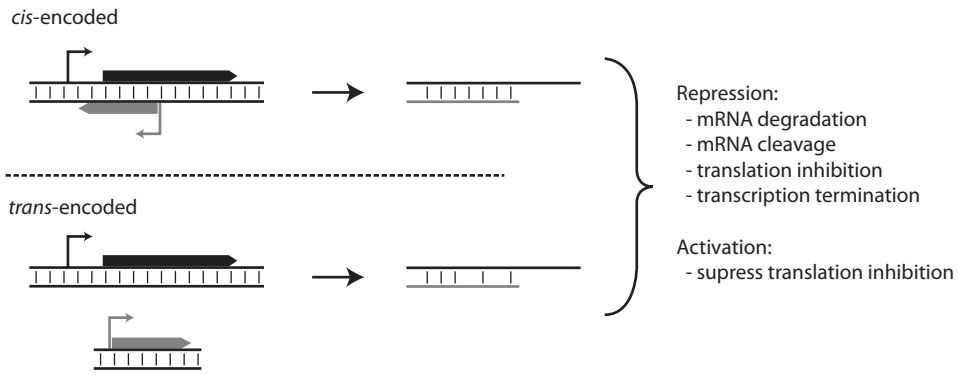


Figure 2.3 Regulation by sRNAs
sRNAs can be either *cis* or *trans*-encoded. The former are transcribed from the DNA strand opposite to their target and, thus, exhibit extensive complementarity (top), whereas the latter are transcribed in DNA regions separated from their target and, therefore, only have limited complementarity (bottom). sRNA binding to target can activate or repress the expression of the target gene via different mechanisms.

2.1.4 Noise in gene expression

The expression of genes is attained by inherently random biochemical reactions that produce the corresponding mRNAs and proteins. As a consequence, differences in phenotype can arise within isogenic (i.e., genetically identical) cell populations subject to constant environmental conditions. The functional relevance of stochastic gene expression was first demonstrated in the genetic circuit underlying the decision between lysis and lysogeny of the phage lambda (Arkin et al., 1998; McAdams and

Arkin, 1997). The authors showed that fluctuations in the concentration of key regulators across a cell population could lead to the activation of different pathways and, hence, explain the observed divergent phenotypes. This observation led researchers to further investigate the importance of gene expression noise in a multitude of biological processes such as metabolism, stress response, development or aging (Raj and van Oudenaarden, 2008).

The protein concentration of a given gene varies from cell to cell in a population. These fluctuations arise from two different sources: 1) the stochastic nature of the processes of transcription and translation (intrinsic noise) (Thattai and van Oudenaarden, 2001); and 2) cell-to-cell variation in the concentration of regulatory molecules, such as RNA polymerases and ribosomes (extrinsic noise) (Elowitz et al., 2002; Swain et al., 2002). The stochastic theory of intrinsic noise predicts that cell-to-cell variability depends on the number of proteins produced per mRNA or the translation efficiency (Thattai and van Oudenaarden, 2001). For example, for two genes being expressed at similar mean abundances, the one with lower mRNA abundance and higher translation efficiency will exhibit larger fluctuations in protein concentration than the gene with lower translation efficiency and higher transcript abundance (Figure 2.4). This happens because proteins are produced in translational bursts and, hence, increased fluctuations in mRNA abundance will lead to increased gene expression noise.

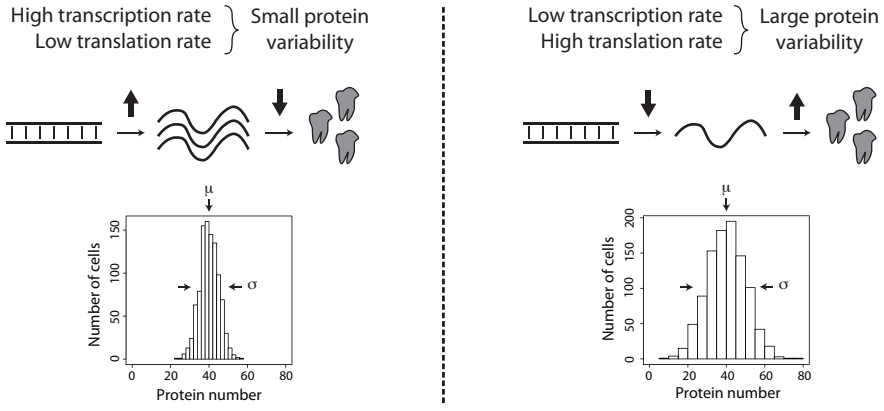


Figure 2.4 Gene expression noise in prokaryotes
For two cell populations with similar average protein level, high transcription rates lead to low protein concentration fluctuations (left panel), whereas low transcription rates and compensating high translation efficiency generate larger variability in gene expression (right panel).

The most common metric to quantify gene expression noise is:

$$\eta\ (noise) = \frac{\sigma}{\mu} \qquad \text{Eq. 2.1}$$

where σ is the standard deviation and μ is the mean of the expression level distribution across a cell population. However, sometimes it is advantageous to use another metric called the noise strength or Fano factor:

$$Fano\ factor = \frac{\sigma^2}{\mu} \qquad \text{Eq. 2.2}$$

The main reason to use the latter metric is because, for a Poisson distribution (assumed for gene expression reactions (Thattai and van Oudenaarden, 2001)), the noise (Eq. 2.1) decreases as the mean increases, whereas the noise strength (Eq. 2.2) is equal to 1 and independent of the mean. Thus, the Fano factor measures the deviation from the theoretical Poisson behavior. This metric is particularly important to quantify the effect of translational burst described above, as this mechanism predicts that noise strength should increase linearly with the average protein abundance when translation efficiency is increased. Practically, this means that if two genes are expressed at a similar average protein level, the one with highest translation efficiency will have larger noise strength and, hence, increased variability (i.e., a broader protein distribution).

2.2 Synthetic biology

Synthetic biology is an emerging research field with a primary goal of making biology easier to engineer. Recent advances in technology have provided the ability to chemically synthesize large segments of DNA, including complete pathways and even genomes (Carr and Church, 2009; Gibson et al., 2010). However, our capacity to understand and design complex genetic circuits is lagging behind (Purnick and Weiss, 2009). To overcome this 'design gap', synthetic biologists postulate that biology should be more like any other engineering field. The main idea is to create genetic sequences encapsulating elementary functions (parts) that can then be assembled together to create higher-order circuits.

2.2.1 Parts, modules and systems

Despite all the advances in the fields of genetic engineering and synthetic biology, the truth is that the successful implementation of synthetic genetic circuits remains a laborious, costly and *ad hoc* research process as a result from multiple rounds of trial-and-error and manual tinkering (Endy, 2005; Kwok, 2010). In 2005, Endy suggested two possible explanations for the slow progress of the field: 1) biology is too complex and we do not understand it completely, hence it may not be amenable for engineering purposes; and 2) biology is only complex because we have never made it simple (Endy, 2005). Indeed, living organisms are the product of millions of years of evolution and, as such, not intended to meet the criteria for human understanding or be the ideal template for engineering purposes.

Nonetheless, natural systems show remarkable modularity — i.e., the parts that compose them can be rearranged in different contexts and still retain their function (e.g., promoters, genes and pathways). These observations have inspired synthetic biologists to seek natural modules whose complexity could be broken down into parts that are easier to understand — hierarchical abstraction (Figure 2.5). As in computer science, abstraction relates to the capacity to encapsulate entities, while hiding all the details implementing their functionality. This would allow researchers to work at any level of complexity without the need to worry about the other levels, but still obey to design rules that enable hierarchical assembly of synthetic biological systems.

Hierarchical abstraction of synthetic biological systems can then be split into: parts, modules and systems. Parts are elementary genetic elements with known biological function such as promoters, genes or terminators. These parts are used as building blocks to attain components of intermediate complexity with a defined function called modules. For example, a module can be a genetic unit composed of a promoter, a 5'UTR, a gene encoding a transcription factor and a terminator. Finally, a system consists in the integration of several modules that work as a whole to implement a useful function. The toggle switch (Gardner et al., 2000) and the repressilator (Elowitz and Leibler, 2000) were pioneer synthetic circuits demonstrating that an engineering-based methodology could also be applied to biology.

A successful realization of the philosophy described above largely depends on our ability to create sets of objects that are orthogonal, reliable, composable and, ideally, tunable. A part/module/system is orthogonal if it does not interfere with functionally related or any other elements in the cell. This property is desirable to ensure that engineered genetic circuits transferred to an organism are not going to impact significantly the physiology of the host, and vice-versa. For example, ribosomes were engineered to recognize mRNA signals different than the ones used by natural *E. coli* ribosomes (Rackham and Chin, 2005).

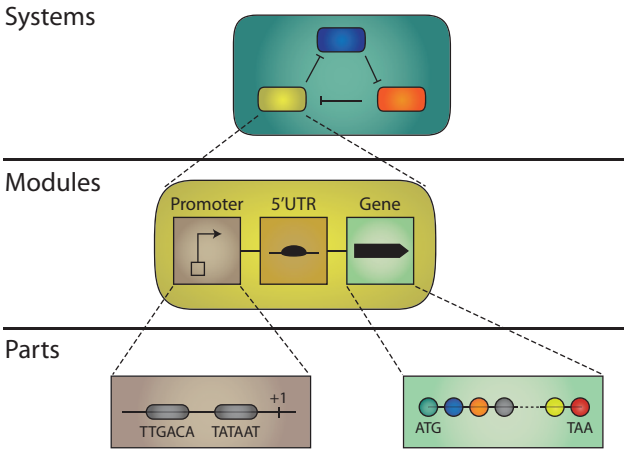


Figure 2.5 Hierarchical abstraction
The development of abstraction layers in biology that can be developed independently will support the engineering of integrated genetic circuits through the predictable assembly of standard biological parts and modules.

In addition, biological objects need to be reliable, i.e., they should work as intended irrespective of the variability offered by their components, host or environment. This property is also intrinsically related to composability, which defines that parts/modules/systems can be combined together without the emergence of any unexpected behavior. Finally, researchers should be able to easily and readily adjust the properties of an object (tunability) to meet their needs. For instance, mutate a specific region of a promoter to attain different transcription initiation rates (Alper et al., 2005). Here, the existence of quantitative sequence-activity models is critical for the rapid and effortless generation of synthetic biological parts on demand (Salis

et al., 2009), as opposed to random mutagenesis strategies that are time-consuming and unpredictable (Anderson et al., 2007; Gardner et al., 2000; Pfleger et al., 2006).

2.2.2 Towards an expression operating system

Genetic circuits link groups of genes so as to execute a coordinated function. The principle mechanism by which coordination occurs is through regulation of gene expression. Therefore, the construction of genetic circuits relies on the availability of parts that control gene expression. Though such parts can be taken from the extraordinary variety of components that nature offers, only a few natural examples are well studied. Thus, a practical approach is to take a well-characterized natural regulatory part and use it as the basis for designing collections of synthetic variants and control different aspects of gene expression. The three basic layers of gene expression are transcription, translation and mRNA degradation.

For transcriptional regulation, the key regulatory parts are promoters (Alper et al., 2005; Deuschle et al., 1986), which determine the frequency by which RNA polymerase initiates transcription of an mRNA, transcription factors (Gonzalez et al., 2010; Zhan et al., 2010; Zhang et al., 2011), which bind promoters to change their activity often in response to effector signals, and *cis*-regulatory RNAs (Lucks et al., 2011), which can abort the continuation of RNA polymerase during transcription.

Once mRNA is produced during transcription, it then proceeds to template the synthesis of protein. Here, another level of regulation, translational control, may occur. At the simplest level, translation initiation is modulated by SD sequences of different strength. Like constitutive promoters, engineered SD sequences of defined strengths offer the ability to set different levels of a gene expression (Barrick et al., 1994), which is particularly useful in the optimization of expression levels in multigene pathways (Wang et al., 2009). In addition, synthetic RNA switches controlling translation are useful elements to encode dynamic circuits where the expression of one gene needs to be linked to another (Isaacs et al., 2004).

Lastly, the stability of an mRNA transcript is also a key determinant of gene expression. Therefore, many parts that control mRNA stability have been constructed (Babiskin and Smolke, 2011; Carothers et al., 2011).

Together, common principles concerning the design and use of elements controlling transcription, translation and degradation are emerging. Standard molecular platforms are being discovered and developed to create families of parts with similar physics of function (Alper et al., 2005; Carothers et al., 2011; Isaacs et al., 2004; Lucks et al., 2011; Zhan et al., 2010) and mapping of sequence to activity (Barrick et al., 1994; Rhodius and Mutalik, 2010; Salis et al., 2009). Each class controls different elements of gene expression dynamics including steady-state levels and timing, with the added functionality of modulating their activity through a sensing layer (Khalil and Collins, 2010). Thus, each may be used to sculpt complex expression control functions of several genes in complex genetic circuits.

2.2.3 Functional composition of parts

Standard biological parts controlling gene expression are meant to be used and reused in different applications. This means that they will be taken out of the context in which they were initially developed and be used to control any gene of interest possibly in specialized host strains and varied culturing conditions. In addition, a part will rarely be used in isolation. More realistically, it will be composed with other regulatory parts into genetic switchboards, complex nodes and regulatory circuits. Therefore, the success of the parts-based approach to genetic circuit engineering depends on the ability of parts to function predictably in novel arrangements.

However, it has become clear that most biological parts are not robust across different contexts. As a result of these context effects, engineering of synthetic biological systems from parts is often reduced to an *ad hoc* process wherein random libraries circuits composed of parts and their variants are screened for desired function (Anderson et al., 2007; Cobb et al., 2012; Pfleger et al., 2006). Thus, many recent efforts in synthetic biology have focused on the development of strategies to overcome the functional variability of parts that prevents predictable assembly. Two divergent strategies can be adopted to ensure the reliable composition of two biological parts: 1) models that predict the deviant behavior arising from divergent context; or 2) development of insulated parts that are insensitive to context change (Figure 2.6).

Achieving accurate models that consider all relevant contextual variables may be utopic. However, for some systems and given set of reasonable constraints, it may become within reach. For example, Salis et al. recently developed a biophysical model to capture the idiosyncrasies of the interactions between the 5'UTR containing a SD sequence and the downstream gene sequence, and predict translation initiation strengths (Salis et al., 2009). This algorithm can further be used to forward engineer new 5'UTR sequences (parts) that attain different levels of initiation. To do that, it does not treat the 5'UTR in isolation, but explicitly models its interaction with surrounding sequences, using the range of sequence 35 nucleotides before and after the start codon as the input. This is predicated on a basic model of translational initiation, where the mRNA unfolds as the ribosome initiation complex is created. In this model, initiation strength depends on the energy required to unfold the sequence upon translational initiation, the energy released from hybridization of the mRNA to the 16S rRNA, the energy released upon initiating tRNA hybridization, the energy of unfolding the standby site, and the energy cost of suboptimal spacing to the start codon. The result is a ribosome binding site (RBS) part evaluator/generator that encompasses a large range of context variations in the several energy terms of its model.

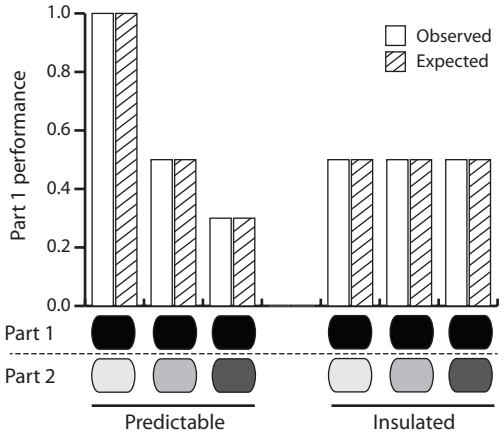


Figure 2.6 Functional composition
The figure depicts the performance of one part (Part 1) when composed with three different parts (Part 2). If a part is insulated from context change, its performance will be constant across the different contexts (right). In contrast, if a part is not insulated, its performance will vary across context (left). In this case, models of part composition are necessary to predict the divergent behavior.

Models like this one are key to the success of the field. They capture the relevant biophysics of the parts and, sometimes, their interactions, and suggest places where enhanced design might eliminate complexities. However, as physical models become incrementally complex, parameterization requires more measurements, which can become excessively costly and time-consuming if these parameters are heavily context-dependent. Therefore, an alternative strategy to make functional composition predictable relies on the ability to insulate parts from surrounding contexts. These techniques mitigate part variability by removing possible interactions with surrounding contexts, thus reducing the number of variables that can affect a part's behavior and simplifying parts-based assembly.

Insulated parts are very appealing since they can be used and reused without worrying about context variability, as opposed to the first strategy where a new part may have to be redesigned every time the context changes (e.g., using the algorithm described above to redesign the 5'UTR every time the gene of interest is altered). Here, it may also look ambitious to assume that insulated parts can be attained for any biological function of interest. Nonetheless, several systems may still be amenable to encode robust insulation. For example, Davis et al. recently created a set of promoters that contained flanking sequences around the minimal ~50 nucleotides encoding a functional promoter in *E. coli* (Davis et al., 2011). The inclusion of ~100 extra nucleotides surrounding the functional promoter resulted in a promoter library that showed less variation than standard promoters across different contexts. For instance, promoters from this set, when placed downstream of specific sequences known to enhance transcriptional activity, were largely insulated from additional stimulation. More recently, the use of transcript cleavage strategies have also shown some success to reliable insulate parts in the mRNA by physically separating them (Lou et al., 2012; Qi et al., 2012).

Ultimately, the combination of reliable parts sets with these effective strategies for their predictable functional assembly and rationally tunable function will result in an efficient design cycle for the construction of complex biological functions.

2.2.4 An engineering and discovery discipline

Synthetic biology has been mainly seen as a discipline to exploit living organisms for human commodity. In fact, some progress have been done in chemical and pharmaceutical production (Keasling, 2010; Lee et al., 2009), therapeutics and diagnostics (Ruder et al., 2011), as well as agricultural and environmental engineering (Zhao and Chen, 2008). For example, the production of an antimalarial drug precursor in engineered yeast was a first step towards reducing of the cost of therapies that, otherwise, would still be unaffordable for most malaria sufferers (Ro et al., 2006). Nonetheless, the application of synthetic biology is not limited to biotechnology applications. A considerable number of synthetic biologists advocate that the engineering-based principles developed can also be used to interrogate and elucidate the basic design principles of natural systems (Bashor et al., 2010; Sprinzak and Elowitz, 2005).

Practitioners in this branch of the field soon adopted the motto 'learn by building'. The main idea is that researchers can learn about the fundamental design principles of biology by iteratively perturbing the system, observing its response and formulating novel hypothesis. Perhaps this philosophy will become clearer by using the example of the repressilator (Elowitz and Leibler, 2000), one of the very first synthetic circuits. This circuit was intended to mimic gene expression oscillations, such as those observed in highly robust cellular circadian clocks. However, this first iteration of a genetic oscillator was far from perfect, namely the oscillatory behavior only worked up to three periods, it was only functional in a fraction of the cells and, lastly, for those cells that were working, the circuit behavior was highly variable. Years later, a new iteration of the genetic oscillator, based on a completely different architecture, supported by computational modeling, was highly stable and tunable (Stricker et al., 2008). More recently, synthetic oscillators have been further optimized to produce synchronized oscillations across the entire cell population via cell-to-cell communication modules (Danino et al., 2010).

The undoubtedly progress of this and other genetic circuits makes clear how iterative engineering attempts can elucidate on general design principles leading to the successful construction of systems of incremental complexity. In fact, such

strategies have also been successful in other fields of engineering, as the history and progress of human-powered flight can attest.

2.3 Computational biology

The genomic era has offered a massive amount of sequence data and greatly contributed to the development of novel computational methods to analyze these large datasets. These tools aim to find and help understand regulatory signals encoded in the DNA sequence and its derivatives molecules: RNAs and proteins. Next, we briefly review two of those methods that are of interest to the present work.

2.3.1 RNA structure prediction

After transcription occurs, nucleotides within a single RNA molecule can base pair with other nucleotides within the same molecule — RNA secondary structure. Within an RNA molecule, A and U, and C and G can form stable Watson-Crick base pairs. In addition, wobble base pairs can also occur, mainly between G and U.

RNA structures are composed of different base pairing patterns: helices, bulges, loops and hairpins (Figure 2.7). Likewise, RNA structures can also be formed between two independent RNA molecules giving origin to a double helix structure. The stability of an RNA structure can be quantified using the amount of free energy released from its base pairs, also known as the Gibbs free energy or ΔG. The more negative the free energy of a structure, the more stable and, hence, likely to be formed.

RNA structures have a wide variety of biological functions and can be predicted using software tools such UNAfold (Markham and Zuker, 2008) or the Vienna RNA package (Markham and Zuker, 2008). Most secondary structure prediction algorithms aim to find the structure with the minimum free energy (MFE). To compute structures of a sequence, empirical determined energy parameters are used. Nonetheless, this does not guarantee that the computationally predicted structure is a real representation of the one occurring *in vivo* since prediction accuracy is still limited (Mathews and Turner, 2006).

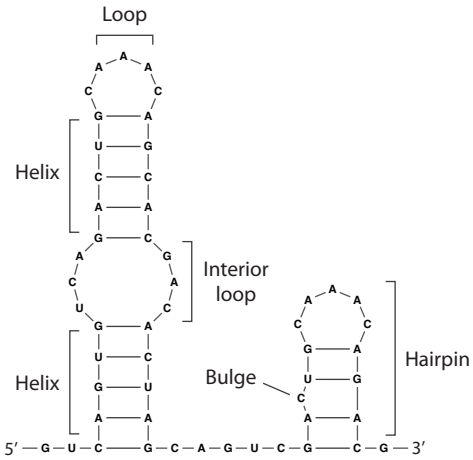


Figure 2.7 RNA structures
An RNA molecule can fold upon itself. RNA structures are composed of multiple base pairing patterns: helix is a run of consecutive base pairs; bulges are single-stranded bases separating two helices; internal loops are facing bulges on both sides of the helix; and loop is a single-stranded region enclosed by a helix. Finally, a run of helices ending in a loop is called a hairpin.

2.3.2 Sequence motifs

Sequence motifs are short and recurrent patterns in DNA that are presumed to encode some biological function. They are especially important when analyzing gene expression regulation, as they can be used to represent extremely relevant sequence patterns such as promoter binding sequences or transcription termination sites.

A common way to describe sequence motifs is through a position frequency matrix (PFM). PFMs, as opposed to consensus sequences that only record the most common element at each position, encode information about the frequency of all the nucleotides/amino acids in each position. Further, we can compute the score of any given sequence with respect to a sequence motif, though this score is not an entirely accurate representation of the reality since it assumes an independent contribution of each position in the motif:

$$score(S) = \sum_i log_2 \left(\frac{f_{S(i),i}}{p_{S(i)}} \right) \qquad \text{Eq. 2.3}$$

where S is the sequence, $S(i)$ is the base at position i in the sequence, $f_{b,i}$ is the frequency of base b at position i in the motif, and p_b is the background frequency of base b in the genome.

Chapter 3

Quantitative Estimation of Activity and Quality for Collections of Functional Genetic Elements

This chapter is based on the article:

Vivek K Mutalik*, **Joao C Guimaraes***, Guillaume Cambray*, Quynh-Anh Mai, Marc Juul Christoffersen, Lance Martin, Ayumi Yu, Colin Lam, Cesar Rodriguez, Gaymon Bennett, Jay Keasling, Drew Endy and Adam P Arkin. (2013). Quantitative estimation of activity and quality for collections of functional genetic elements. *Nature Methods*, *in press.*

(* equal contribution)

3.1 Abstract

The practice of engineering biology now depends on the *ad hoc* reuse of genetic elements whose precise activities vary across changing contexts, thereby making design of individual DNA sequences encoding specific phenotypes difficult. Methods are lacking with which researchers can affordably coordinate the quantification and analysis of part performance across varied environments, as needed to identify, evaluate, and improve problematic part types. We developed an easy-to-use analysis of variance (ANOVA) framework for quantifying the performance of genetic elements controlling gene expression. As proof of concept, we assembled and analyzed combinations of prokaryotic transcription and translation initiation elements driving the expression of two sequence-divergent fluorescent reporters. We quantified to what extent the interactions between transcription and translation initiation elements, and downstream genes leads to unpredictable gene expression levels. We also determined how estimation of part activity relates to the number of unique element combinations tested, and show that measurement strategies can be optimized given finite measurement resources, including how to estimate expected ensemble-wide part activity from just one or two measurements. We propose a new characteristic, biomolecular part "quality," that is defined as the quantitative variation in part performance across changing contexts. Quantitative measures of primary part activities, variation in activities across changing contexts, and functional coupling across encoded element boundaries are needed to collectively improve the quality of engineered genetic elements via distributed and asynchronous research efforts.

3.2 Introduction

Genetic engineers have traditionally worked in isolation to implement stand-alone systems that require the expression of a small number of heterologous genes (Keasling, 2010; Wilkinson and Micklefield, 2007). For a small but increasing number of applications, engineers must a priori specify required biomolecular activities with quantitative precision (Anderson et al., 2006; Bonnet et al., 2012; Chen et al., 2010; Dubendorff and Studier, 1991; Mertens et al., 1995; Saeidi et al., 2011;

Widmaier et al., 2009; Xie et al., 2011). As capacities to synthesize and assemble DNA continue to increase geometrically (Carr and Church, 2009), practitioners are further challenged to develop frameworks for designing DNA sequences that might collectively encode and precisely express hundreds of coding sequences (Cambray et al., 2011; Cardinale and Arkin, 2012).

Two societal factors are also changing the practice of genetic engineering. First, distributed research communities have emerged whose members work to collectively implement collections of standard biological parts (Canton et al., 2008; Smolke, 2009). Scalable frameworks that enable shareable quantitative descriptions of part activities and quality are needed to improve such work. Second, some applications of synthetic biology are being developed for use in humans or in minimally contained environmental settings (Anderson et al., 2006; Ruder et al., 2011; Saeidi et al., 2011; Sinha et al., 2010; Xie et al., 2011). While other "public works"-focused engineering disciplines (e.g., structural engineering) have or are adapting probabilistic methods enabling design of systems via analysis of performance limit states (Gulvanessian and Holicky, 2005), bioengineers are not yet able to reliably estimate expected system performance distributions from quantitative descriptions of molecular elements (Cambray et al., 2011; Endy, 2005).

Decades of sustained research has focused on characterizing the "strengths" or "activities" of functional genetic elements. For example, many studies have considered gene expression control elements within natural systems (Hook-Barnard and Hinton, 2007; Mutalik et al., 2009; Shimada et al., 2004). Such work reveals that the quantitative activity of genetic elements can be highly context dependent (Cho and Yanofsky, 1988; de Smit and van Duin, 1994; Ellinger et al., 1994; Yarchuk et al., 1992). Engineers and biologists alike have used such foundational knowledge to generate and study libraries of synthetic expression control elements (Alper et al., 2005; Barrick et al., 1994; Carrier and Keasling, 1999; Cox et al., 2007; de Smit and van Duin, 1990; Ellis et al., 2009; Isaacs et al., 2004; Mutalik et al., 2012; Salis et al., 2009). So-produced collections are widely used in applications despite being characterized on an *ad hoc* basis and across a relatively limited range of contexts. In some cases, researchers have used first-principle physical chemistry models to develop predictors of element function that attempt to account for context impacts (de Smit and van Duin, 1994; Salis et al., 2009; Yager and von Hippel, 1991). How-

ever, while valuable, these models are not yet able to fully capture the impact of changing contexts on genetic element function. More recently, researchers have developed passive and active genetic insulators such that the functioning of one genetic element might not corrupt the functioning of a neighboring element (Babiskin and Smolke, 2011; Davis et al., 2011; Lou et al., 2012; Qi et al., 2012). However, lacking systematic and quantitative data detailing how and to what extent different types of genetic elements interact, it is unclear if these projects have focused on regularizing the most difficult element-element junctions.

Here, we sought to develop an easy-to-deploy mathematical framework that can be used to score the intrinsic activities of genetic elements and also track how such activities vary (or not) across changing contexts. We propose that the latter metric can be used as a quantitative score for how the activities of parts vary when reused in combination with other parts or across changing conditions and that can serve a public (i.e., shared) measure of part "quality." To develop and demonstrate such a method, we chose to evaluate transcription and translation elements commonly used to regulate gene expression in *E. coli*. We constructed a full combinatorial expression library from frequently used transcription and translation elements to express two sequence-distinct genes and measured individual levels of mRNA and protein for all element combinations. We then quantified element activity and quality using the full factorial experimental dataset. The resulting scores enable rational selection of elements based on specified performance and quality requirements and identify the junction between translation initiation elements and downstream genes as the major source of irregular gene expression levels. We also demonstrate that only a few measurements are necessary to estimate the activity of new elements to within reasonable accuracy once a baseline combinatorial mapping is established.

3.3 Materials and methods

3.3.1 Bacterial strains, plasmids and growth conditions

Detailed information on sequence designs, plasmid maps and corresponding experimental datasets for each construct developed in this study are available in a public repository database at http://biofab.org/data.

The Promoter:5'UTR combinatorial library driving the expression of *sfgfp* (Pedelacq et al., 2006), hereafter *gfp*, and *mrfp1* (Campbell et al., 2002), hereafter *rfp*, was carried on medium copy vectors and assembled using the Golden Gate cloning method (Engler et al., 2008). A subset of the library driving the expression of *gfp* comprising each of the seven promoters combined to at least six of the eleven 5'UTR was inserted on the chromosome at the main target attachment site of the phage lambda (*attBλ*) to assess the gene copy number effect.

E. coli strain BW25113 was used for plasmid construction purposes and for fluorescence measurements. All strains were grown in MOPS EZ rich media (Teknova) and all the experiments were conducted in triplicate (replicate assays from independent overnight liquid cultures).

3.3.2 *In vivo* assays using the plate reader and flow cytometer

Cultures were grown in 2 ml 96 deep well plates containing 400 µl of MOPS EZ rich media (Teknova, cat.# M2105) with appropriate antibiotics, inoculating 3 µl from thawed glycerol stocks. Cultures were grown overnight in 96 well plates at 37 °C with shaking at 900 rpm on Multitron shaker (Inforys-HT).

For microplate kinetic assays, overnight cultures were diluted 1:50 into a final volume of 150 µl of fresh media with appropriate antibiotics in clear bottom black plates and incubated in a multimode microplate reader-incubator-shaker Synergy-2 (BioTek Instruments Inc.). Cultures were grown for 6 hours with rapid shaking and repeated measurements for optical density at a wavelength of 600 nm (OD_{600}) and fluorescence (relative fluorescence units or RFU) were performed every 10 minutes.

For the flow cytometer assays, overnight cultures were diluted 1:50 into a final volume of 200 µl fresh media with appropriate antibiotics in 96 deep well plates and grown for 2 hours (to exponential phase with OD_{600} in the range of 0.3 to 0.5 in microplate reader) at 37 °C with shaking at 900 rpm on Multitron shaker. Cultures were diluted 1:2,000 in chilled and filtered phosphate buffered saline (PBS) (pH 7.4) containing 500ug/ml streptomycin in chilled 96-well clear plates (Costar) and immediately subjected to flow cytometer analysis. We used a Guava® EasyCyte flow

cytometer (EMD Millipore, Hayward, CA) equipped with microcapillary and auto-sampling capabilities. For each sample, ~2,000 cells were collected.

3.3.3 Transcriptional analysis by qPCR (quantitative polymerase chain reaction)

Cultures were prepared and grown as described above for 2 hours, to reach exponential phase with OD_{600} in the range of 0.3 to 0.5 when measured in microplate reader. After 2 hours, cultures were harvested on ice and total RNA was extracted by enzymatic lysis with lysozyme, followed by β-mercaptoethanol and ethanol treatment. Total RNA concentration in each sample of 96-well plate was quantified using a Nanodrop 1000 (Thermo Scientific), and a volume corresponding to 25 μg of total RNA was used for qPCR. Using the Power SYBR Green RNA-to-C_T 1-Step Kit (Applied Biosystems), reverse transcription of RNA standard and samples were performed and immediately followed by qPCR in a 1-step reaction in Applied Biosystem's StepOnePlus instrument (manufacturer's protocol). The transcript abundances of reporter genes *gfp* and *rfp* were quantified using a standard curve.

3.3.4 Plate reader kinetic assay data analysis

Background fluorescence of cultures was determined using a combinatorial library of the seven promoters combined with a non-functional 5'UTR ("dead RBS" 5'UTR) with each of the two reporters in *E. coli* BW25113. These control strains were grown and assayed along with each of the combinatorial libraries on the same 96-well plates. Their fluorescence signals were averaged to generate a standard curve for OD against relative fluorescent units. The standard curve was used to subtract background fluorescence from the reporter strain fluorescence value at the same OD, yielding a background subtracted OD vs RFU differential rate plot for each strain carrying each member of the combinatorial library. The slope of the linear portion of each differential rate plot was taken as exponential steady state fluorescence (f_{ss}). To account for the growth rate and the maturation rate constant for the stable fluorescent protein, we used a previously published model (Leveau and Lindow, 2001):

$$Expression\ strength = f_{ss} \times \mu \times \left(1 + \frac{\mu}{m}\right) \quad \text{Eq. 3.1}$$

where μ is the culture growth rate, m is the fluorescent reporter maturation constant and f_{ss} is the steady state fluorescence. The protein maturation rates for green fluorescent protein (GFP) and red fluorescent protein (RFP) were obtained from the literature (~2.77 h^{-1} for GFP (Iizuka et al., 2011) and ~1 h^{-1} for RFP (Campbell et al., 2002)) while both steady state fluorescence and growth rate (h^{-1}) were experimentally determined for each candidate in the library.

3.3.5 Flow cytometer data analysis

For each replicate the FCS (flow cytometry standard) files were parsed and analyzed using in-house scripts for R software (http://www.r-project.org/). In addition to the gating parameters defined at the time of acquisition, a custom automated gating procedure was developed to maximize consistency in the results (Lo et al., 2009). Namely, the data were clustered on the appropriate log-fluorescence signal, allowing for 1 or 2 clusters. This step was used to control the quality of the data through the identification of well-to-well contaminations or selection for loss of function mutants, both of which occasionally occurred during experiments. A specific criterion combining the integrated completed likelihood (ICL) and the Bayesian information criterion (BIC) for the fitted mixture model was applied to determine the presence of bimodality in the fluorescence data. We used this strategy to flag experiments that needed to be redone. Finally, for the filtered data set, the average and variance of the linear fluorescence data was calculated from the cell population.

3.3.6 ANOVA framework

To account for differences in the fluorescence intensities of reporters and in qPCR primer efficiency, we mean-centered our datasets for each gene of interest (GOI) independently. Then, using three replicates of fluorescence, transcript abundance, and translation efficiency we performed ANOVA (Miller, 1986) on the respective linear models (below) using the 'anova' routine in R software.

The main effects (the primary scores for Promoters, 5'UTRs and GOIs) were directly retrieved from the ANOVA table of effects (accessed using the 'model.tables'

function in R). In essence, scores are calculated as the mean of all observations comprising a given level of a factor to which the grand mean (α) is subtracted (e.g., the mean of all observations containing p1 promoter subtracted by the mean of all observations yields the primary score for p1). To ease visualization and interpretation, the grand mean (α) was distributed over the main effects so that all resulting scores are positive.

The integrated deviation from the main effect (secondary scores) for each element resulting from its composition with different parts, was calculated as the standard error of the mean (SEM) of the respective interaction terms effects (e.g., the context sensitivity of p1 due to composition with different UTRs was calculated using the SEM of the set {p1:u1 effect, p1:u2 effect, ..., p1:u11 effect}). By definition, the sum of these interaction terms equals zero. Hence, the SEM effectively measures the spread of the interaction around main effects.

3.3.7 Accuracy of estimated scores for new parts

To evaluate the accuracy of our model estimating the scores for a new Promoter and/or 5'UTR (i.e., for elements not included in the training model), we performed a cross-validation analysis using our entire dataset. Here, the goal is to determine how the robustness of an estimated score for a new Promoter (or 5'UTR), with the reporter kept constant, changes as we characterize this new Promoter (or 5'UTR) with an increasing number of 5'UTRs (or Promoters). Please note that here we define the "true" score of a Promoter (or 5'UTR) as being its estimated score across all combinations (i.e., across the entire dataset).

In detail, the following steps were taken to determine the accuracy of the estimated score for a new Promoter by a cross-validation approach:

1. Create different combinations of 5'UTRs with varied number of elements ranging from 1 to 11 (e.g., (u1),...,(u11), (u1,u2),...,(u10,u11), (u1,u2,u3),...)
2. For each of combination of 5'UTRs defined in (1):

2.1. Estimate the score of each of the 7 promoters in the dataset using the experimental data corresponding only to the usage of those promoters with the 5'UTRs elements present in that combination and based on the linear model described in Eq. 3.5.

2.2. Calculate the absolute error (AE) as the difference between the Promoter score estimated in (2.1) and the "true" score of the Promoter (estimated with all combinations of 5'UTR).

2.3. Calculate the relative error (RE) by dividing the AE computed in (2.2) by the "true" score of the Promoter.

2.4. Average all the relative errors across all the promoters to produce an aggregated metric for that number of test 5'UTR combinations. That is, for every 5'UTR group defined in (1), we calculate seven REs (because we have seven Promoters) and average them all to produce a single metric.

An equivalent analysis was carried out to determine the accuracy of estimating the score of a new 5'UTR as a function of the number of combinations of Promoters tested.

3.3.8 Evaluation of temperature effect on part performance

The gene expression model developed here can be further extended to consider any variable of interest by simply adding the new factor and corresponding interactions to the linear regression. Therefore, to estimate the effect of temperature on the performance of transcription and translation elements (Promoter and 5'UTR) we consider the following extended linear model for ANOVA, where T is the temperature, P is a promoter element and U is a 5'UTR element (see Eq. 3.5 for details):

$$\log(F_{ijk}) = \alpha + P_i + U_j + T_k + (P{:}U)_{ij} + (P{:}T)_{ik} + (U{:}T)_{jk} + (P{:}U{:}T)_{ijk} + \varepsilon_{ijk} \quad \text{for } i = \{1 \text{ to } 7\};\ j = \{1 \text{ to } 11\};\ k = \{1, 2\} \qquad \text{Eq. 3.2}$$

3.4 Results

3.4.1 Quantifying context effects as a score of part quality

We first considered a conventional model (Klumpp et al., 2009) to represent population average steady state protein expression levels from a constitutive promoter and translation initiation element:

$$PA = g \times T_x \times T_r \,/\, (k_d \times k_m) \qquad \text{Eq. 3.3}$$

$$\log(PA) = \log(g) + \log(T_x) + \log(T_r) - \log(k_m) - \log(k_d) \qquad \text{Eq. 3.4}$$

where, PA is a steady state protein abundance, g is the gene copy number, T_x the transcription rate per gene copy, T_r the translation rate per mRNA, and k_m and k_d are the mRNA and protein degradation rates, respectively.

To adopt this model (Eq. 3.3) to the individual activities of multiple genetic elements, we would have to assume that those activities were fixed across different contexts and that each element could be uniquely mapped to the described parameters. However, if elements are reused in novel combinations or across varying operational contexts then their activities can change. We thus developed a linear model inspired from Eq. 3.4 to enable analysis of element-element context effects:

$$\log(O_{ijk}) = \alpha + P_i + U_j + GOI_k + (P{:}U)_{ij} + (P{:}GOI)_{ik} + (U{:}GOI)_{jk} + (P{:}U{:}GOI)_{ijk} + \varepsilon_{ijk} \qquad \text{Eq. 3.5}$$

$$\text{for } i = \{1 \text{ to } 7\};\ j = \{1 \text{ to } 11\};\ k = \{1 \text{ to } 2\}$$

where O_{ijk} is an output signal (arbitrary fluorescence level, transcript abundance, or translation efficiency — defined as the protein fluorescence divided by mRNA abundance) measured for a genetic construct comprising the i^{th} Promoter (P_i), j^{th} 5' UTR (U_j) and k^{th} gene of interest (GOI_k), while $(P{:}U)_{ij}$, $(P{:}GOI)_{ik}$ and $(U{:}GOI)_{jk}$ represent pair-wise context effects among elements, $(P{:}U{:}GOI)_{ijk}$ captures effects specific to a given combination, and α is the average signal. We estimated an error term for each combination, ε_{ijk}, to quantify any remaining unexplained variation.

Although functional relationships among genetic elements and changing environments will not always map to such simple, molecular mechanism-agnostic linear models, our underlying goal was to establish a basic framework that many labs might easily use to contribute to the estimation of activities and quality of biological parts, and to share such information in working to improve part collections. Stated differently, the framework developed above is primarily focused on the recording and reporting of measurements and not deeper mechanistic understanding. By improving a collective capacity to identify categories of low quality parts and problematic element-element junctions, we seek to enable, prioritize, and evaluate subsequent work to better understand and ultimately engineer higher quality genetic elements.

3.4.2 Experimental design

Many extrinsic factors can overwhelm observed variation in the activities of transcription control and translation initiation elements. Thus, a first challenge was to determine if we could directly observe subtle or modest quantitative variation in genetic element activities arising only from the reuse of parts in combination. To do so we first pursued carefully controlled repeat experiments under common physical conditions.

We selected widely used, representative genetic elements encoding transcription control and translation initiation functions. While we hereafter refer to each category of control elements as "Promoters" or "5'UTRs" we note that our selected elements are encoded by irregular DNA sequences as reported and as typically used elsewhere. For example, "Promoters" include DNA sequence beyond the transcription start that would contribute promoter-associated mRNA sequence to any coupled 5'UTR and thereby potentially modulate both translation initiation and mRNA stability. Moreover, the DNA sequence after a transcription start site is also known to modulate RNA polymerase promoter escape and hence could also affect promoter strength (Hook-Barnard and Hinton, 2007). As specific context, the total number of nucleotides (nts) preceding the translation initiation codon varies from 21 to 59 nts across the mRNA encoded by the Promoters and 5'UTRs assembled here (Figure 3.1).

We constructed a full combinatorial library of seven Promoters and eleven 5'UTRs upstream of two distinct genes of interest (*gfp* and *rfp*; 52% nucleotide identity overall; 56% over the first 30 codons) and a common 3'UTR context (Figure 3.1). We placed each expression construct within a medium copy number plasmid and also integrated a subset of element combinations into the bacterial chromosome (Materials and methods). Then, the expression levels of the different constructs were monitored via repeated measurements of steady state fluorescence for both GFP and RFP proteins, and also via the abundance of individual transcripts using qPCR (Materials and methods). Comparing each measurement type enabled the individual contributions of transcription and translation processes to be estimated.

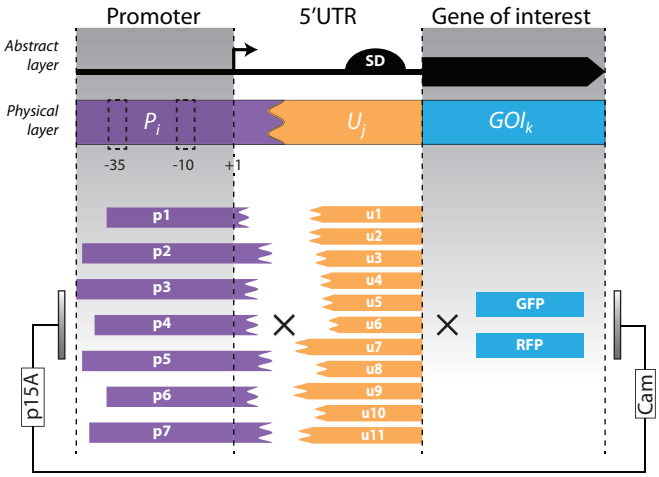


Figure 3.1 Composition of irregular transcription and translation genetic elements
Schematic of Promoter (P) and 5'UTR (U) elements assembled in combination here. While genetic elements are typically represented in tools, models, and simulations as well-defined functionally independent parts (Abstract layer), practiced genetic engineers typically use elements having unique boundaries and thus a high potential for functional couplings across elements (Physical layer).

3.4.3 Measurement of Promoter-5'UTR combinations

We quantified gene expression across the library by measuring fluorescence and mRNA levels under defined growth conditions (Figure 3.2A-D). Bulk culture fluorescence and growth profiles were monitored over time using an automated fluorimeter, mRNA levels were measured by qPCR, and single-cell fluorescence distributions by flow cytometry at a single time point during exponential growth (Materials and methods). Single cell and growth normalized bulk culture measurements of fluorescence were highly correlated (coefficient of determination $R^2 = 0.96$ for both GFP and RFP libraries) and exhibited high day-to-day reproducibility in triplicate experiments (average $R^2 \sim 0.98$). Fluorescence measurements were also well correlated between plasmids and chromosomal integrants ($R^2 = 0.85$). Once we established that no element combination produced atypical behavior (e.g., altered growth rate or cell-cell expression heterogeneity), we used population average fluorescence levels as determined by cytometry for our following analyses.

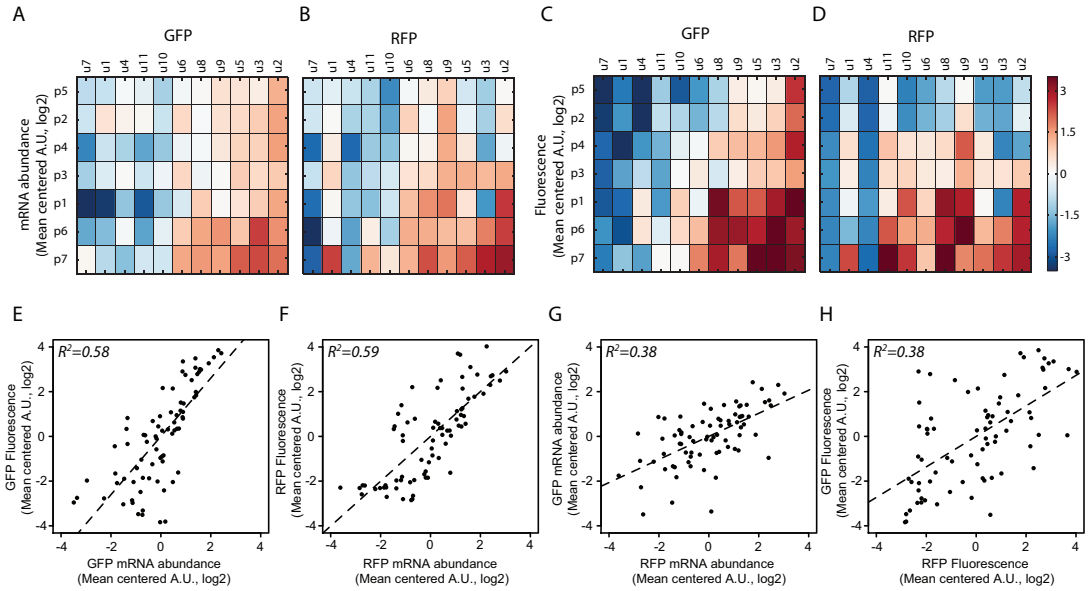


Figure 3.2 Observed variation and correlation of mRNA abundance and protein fluorescence from combinatorial library of expression control elements

(A-B) Heatmaps showing mRNA abundance for all combinations of transcription (p_i, rows) and translation (u_j, columns) elements driving the expression of *gfp* (A) or *rfp* (B). Each value is a dimensionless number corresponding to mean mRNA abundance measured from a cell population by bulk qPCR divided by the average abundance for all constructs within that panel. (C-D) Similarly mean-centered values for population average fluorescence intensities as measured by flow cytometry. The order of the elements in the matrices corresponds to a bi-dimensional clustering performed on the data in panel C and held constant to facilitate visual comparison. Abundances are expressed on a $\log_2$ scale (mean centered arbitrary units — A.U.) and colored according (thermometer scale). (E-F) Scatter plot of mRNA abundance versus fluorescence for constructs driving *gfp* (E) and *rfp* (F) expression. (G-H) Pair-wise comparison between mRNA levels (G) and fluorescence (H) for constructs driving *gfp* and *rfp*.

3.4.4 Variation and correlation of observed expression levels

Fluorescence values measured across the library varied over a 206- and 117-fold range for GFP and RFP, while mRNA levels varied over a 542- and 354-fold range, respectively. Protein and transcript abundance data indicated that a few Promoters and 5'UTRs elements encoded a consistent impact on expression across multiple combinations (Figure 3.2A-D). Additionally, we observed some non-systematic variation with specific combinations of Promoters and UTRs across the two different reporters, indicating more complex interactions among parts. For example, the p1:u1 combination produced ~11-fold more *rfp* than *gfp* mRNA and ~6-fold more

red than green fluorescence. In contrast, the p1:u3 combination produced ~7-fold more *gfp* than *rfp* mRNA and ~37-fold more green than red fluorescence.

A pair wise comparison of transcript abundance and fluorescence levels for each combination of control elements across the two reporters indicated that measured mRNA values account for ~60% of the total variation in fluorescence levels (R^2 = 0.58 and R^2 = 0.59 on log-log scale for GFP and RFP libraries, respectively; Figure 3.2E-F). Pair wise comparison of transcript abundances (R^2 = 0.38, Figure 3.2G) and fluorescence levels between the two reporter libraries revealed more modest correlations (R^2 = 0.38, Figure 3.2H).

3.4.5 Quantifying the performance of parts

We used a linear log-transformed model of gene expression (Eq. 3.5) to quantify the individual contributions of Promoters, 5'UTRs, and GOIs to different expression phenotypes (fluorescence, mRNA level, or translation efficiency), as well as to quantify interactions among elements. More specifically, we conducted a full factorial analysis of variance (ANOVA) with repetitions (Miller, 1986) to predict the output of the system based on the identity of each element and element-element interactions as instantiated in any given construct.

We first sought to quantify how each category of genetic element and interactions among elements contribute to differences in expression levels. We assumed that the specific Promoters, 5'UTRs, and GOIs used here define representative samples for each element type and used a random effect interpretation of the ANOVA results. From this assumption, we estimated the overall contribution of each element type and type-type interaction to expression levels (Figure 3.3). We used mean-centered transcript abundances and fluorescence levels to remove confounding effects arising from systematic biases in experimental signals between *gfp* and *rfp* reporters (Materials and methods) while preserving interaction factors among the control elements and coding sequences themselves (Table 5, Table 6 and Table 7). Not surprisingly, we found that the 5'UTRs and Promoters are the major contributors to variation in expressed fluorescence levels (46% and 37%, respectively; Figure 3.3A). Also expected but quantified here, interaction across 5'UTR:GOI junctions accounted for ~14% of total variation, whereas the combined contribu-

tions of all remaining interaction effects were almost negligible (< 4% combined). Further analysis found 5'UTR identity to be the dominant factor (59%) in determining mRNA abundance, followed by the Promoter (21%). Again, 5'UTR:GOI interactions demonstrated an important contribution (9%) to mRNA abundance (Figure 3.3B and Table 6). For translation efficiency, surprisingly, Promoter identity emerged as the key factor (54%), followed by the 5'UTR (25%) and 5'UTR:GOI interactions (14%) (Figure 3.3C and Table 7). The remaining error (ε) was the least important factor for all three experimental data types (fluorescence, mRNA, and translation efficiency) corresponds to the experimental error.

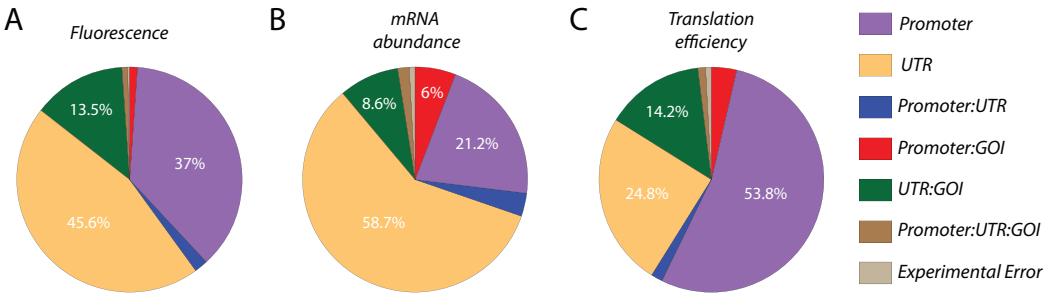


Figure 3.3 Quantification of factors and interactions contributing to variation in mRNA abundance, translation efficiency, and gene expression

Full factorial ANOVA was conducted to quantify the average contributions from genetic element types, as well as interactions among elements, to total variation in gene expression levels (A), mRNA abundance (B) and translation efficiency (C).

Table 5 ANOVA table for fluorescence

Sum of squares represent the actual explanation of variation in the output measurement. The mean squares represent the average contribution of each of the factors/interactions taking into account their degrees of freedom (df). Output variables were mean centered per GOI to control for the signal bias between the two reporters (column mean sq. given GOI).

	df	Sum sq.	Mean sq.	Mean sq. given GOI	%
Promoter	6	421.3	70.2	70.2	37.0
UTR	10	865.7	86.6	86.6	45.6
GOI	1	5.0	5.0	0.0	0.0
Promoter:UTR	60	227.7	3.8	3.8	2.0
Promoter:GOI	6	13.1	2.2	2.2	1.2
UTR:GOI	10	255.5	25.6	25.6	13.5
Promoter:UTR:GOI	60	98.7	1.6	1.6	0.9
Residuals	308	15.9	0.1	0.1	0.0

Table 6 ANOVA table for mRNA abundance
Sum of squares represent the actual explanation of variation in the output measurement. The mean squares represent the average contribution of each of the factors/interactions taking into account their degrees of freedom (df). Output variables were mean centered per GOI to control for the signal bias between the two reporters (column mean sq. given GOI).

	df	Sum sq.	Mean sq.	Mean sq. given GOI	%
Promoter	6	96.3	16.1	16.1	21.2
UTR	10	444.2	44.4	44.4	58.7
GOI	1	620.8	620.8	0.0	0.0
Promoter:UTR	60	141.8	2.4	2.4	3.1
Promoter:GOI	6	27.1	4.5	4.5	6.0
UTR:GOI	10	64.9	6.5	6.5	8.6
Promoter:UTR:GOI	60	76.3	1.3	1.3	1.7
Residuals	308	191.0	0.6	0.6	0.8

Table 7 ANOVA table for translation efficiency
Sum of squares represent the actual explanation of variation in the output measurement. The mean squares represent the average contribution of each of the factors/interactions taking into account their degrees of freedom (df). Output variables were mean centered per GOI to control for the signal bias between the two reporters (column mean sq. given GOI).

	df	Sum sq.	Mean sq.	Mean sq. given GOI	%
Promoter	6	292.2	48.7	48.7	53.8
UTR	10	225.0	22.5	22.5	24.8
GOI	1	737.0	737.0	0.0	0.0
Promoter:UTR	60	93.0	1.6	1.6	1.7
Promoter:GOI	6	19.9	3.3	3.3	3.7
UTR:GOI	10	128.6	12.9	12.9	14.2
Promoter:UTR:GOI	60	56.0	0.9	0.9	1.0
Residuals	308	223.6	0.7	0.7	0.8

We next sought to quantify the primary activity of individual elements. To do that, we used a fixed effect interpretation of the ANOVA results to estimate part-associated scores that characterize the average performance for any given genetic element and a set of sub-scores quantifying the variability in performance arising from interactions among elements (Figure 3.4). Using the fluorescence data, we first estimated the main effect of each Promoter, 5'UTR, and GOI to capture the average contribution of a given element to expression levels across all genetic contexts in which it was found (Material and methods). We used these statistics to define a "primary score" for each element (Figure 3.4, main bars). Primary scores must

be corrected by appropriate interaction term(s) to yield adjusted estimates of expression for a given combination of elements (Eq. 3.5).

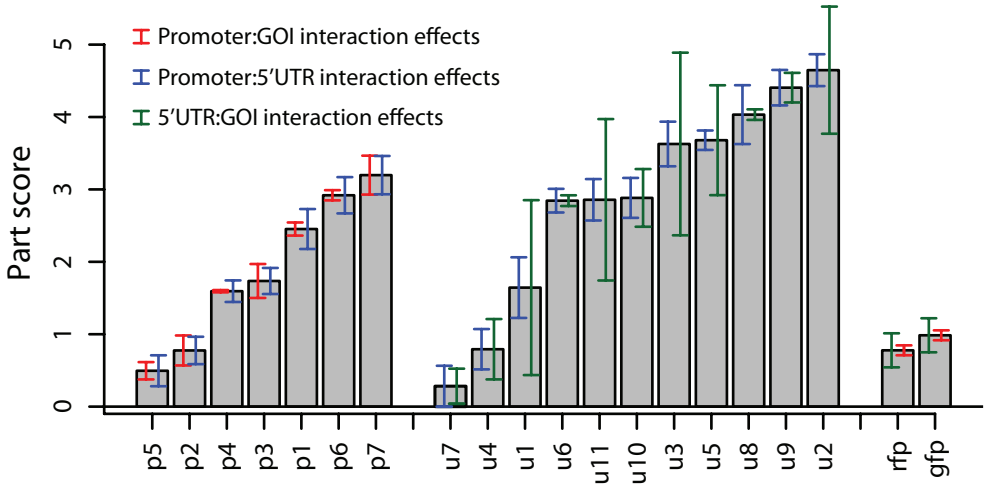


Figure 3.4 Performance and quality scores for genetic elements
Primary part activity scores (bar height, $\log_2$) giving the relative contribution of each Promoter (p_i), 5'UTR (u_j), and gene of interest to observed fluorescence. Error bars indicate the standard error of all interactions involving each element with all other elements in a different functional category. As such, they reflect the variation of element performance in response to changes in proximal genetic context. Reciprocal interactions are color-coded as follows: transcription elements and GOIs (red), transcription and translation elements (blue), and translation elements and GOIs (green).

We then quantified variation in individual element activities across changing contexts as realized here by making many part-part combinations. We grouped element-element interactions by functional category and computed a set of secondary scores that define the sensitivity of a given element to each context variable (Figure 3.4, error bars). The smaller a secondary score, the more likely an element will maintain its primary activity across different contexts (i.e., higher quality); conversely, larger secondary scores indicate greater context dependency (i.e., lower quality). Of note, elements with similar individual primary scores can have different secondary scores. For example, the u6, u11, and u10 5'UTRs all have a similar average primary activity (~3), but different corresponding secondary scores with respect to changing coding sequences (u11>u10>u6, green error bars, Figure 3.4). In more detail, the primary activity of u11 is strongly influenced by GOI identity (green error bars) and, to a lesser extent, by promoter identity (blue error bars). In contrast, the activity of u6 is largely insensitive to changes in either adjacent genet-

ic element. Such information enables selection of elements for use in optimal testing strategies (below) and in designing synthetic genetic systems.

The 5'UTR:GOI junction was the major source of interactions between the gene control elements tested here. To further identify what were the emergent properties arising from the different combinations that could justify this effect, we sought to evaluate the influence of RNA secondary structures formed when the same 5'UTR is used in combination with the different GOIs (de Smit and van Duin, 1990, 1994). As expected, we observed that the RNA structure stability — defined by the minimum free energy (MFE) — varies with the different 5'UTR:GOI combinations (Figure 3.5A). We also found a good correlation between the variation of 5'UTR score (ΔScore) and the change in the MFEs (Pearson correlation $r \sim 0.8$, Figure 3.5B).

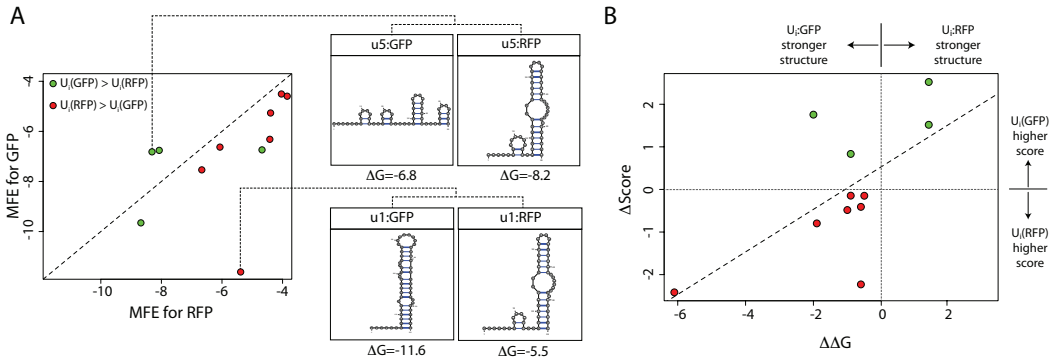


Figure 3.5 Variability in 5'UTR scores is correlated with RNA structure at the 5'UTR:GOI junction. To evaluate the source of variability of 5'UTR scores, we used UNAfold software (Markham and Zuker, 2008) to predict RNA structure formed at the junction between all the 11 5'UTRs and 2 GOI ([-19,31] with respect to AUG). We observed that different RNA structures are formed when the same UTR is used with different GOIs (A). For a 5'UTR, the weaker the structure formed the higher is expected to be its activity. For example, if the structure formed with GFP is weaker than the one formed with RFP (space above dashed line), the score of the 5'UTR fused with GFP is expected to be higher than when fused with RFP. 5'UTRs driving stronger GFP expression than RFP are colored in green, while those driving stronger RFP expression than GFP are colored in red. Using this qualitative analysis, we find that 9 out of the 11 UTRs in our dataset have the expected behavior. (B) The difference between the folding energies of GFP and RFP (ΔΔG) are well correlated with the difference in 5'UTR performance between GFP and RFP (ΔScore) ($r = 0.77$). When ΔScore > 0, it means that UTR performance is higher with GFP than RFP (green dots), conversely if ΔScore < 0, it means that UTR performance is higher with RFP than GFP (red dots). UTRs that have stronger structures when fused with GFP than with RFP have ΔΔG < 0 and UTRs that have stronger structures with RFP than GFP have ΔΔG > 0.

3.4.6 Framework extension to extrinsic environmental factor

We repeated GFP-only expression measurements in cultures grown at 30 °C to test if our framework could be extended to factors other than genetic context (Materials and methods). We found that GFP expression levels from the full combinatorial library were well correlated between 30 °C and 37 °C culture conditions ($R^2 \sim 0.97$, Figure 3.6A). Overall, changes across this temperature range accounted for less than 0.5% of total observed variation in GFP expressed from the given elements (Figure 3.6B-C), suggesting that the selected elements do not encode element or junction-specific structures responsive to this temperature difference per se. We also noted that the overall Promoter:5'UTR interaction estimated from observed variation in 30 °C and 37 °C GFP levels was the same as that estimated from the observed variation in 37 °C only (~2%, Figure 3.3 and Figure 3.6B).

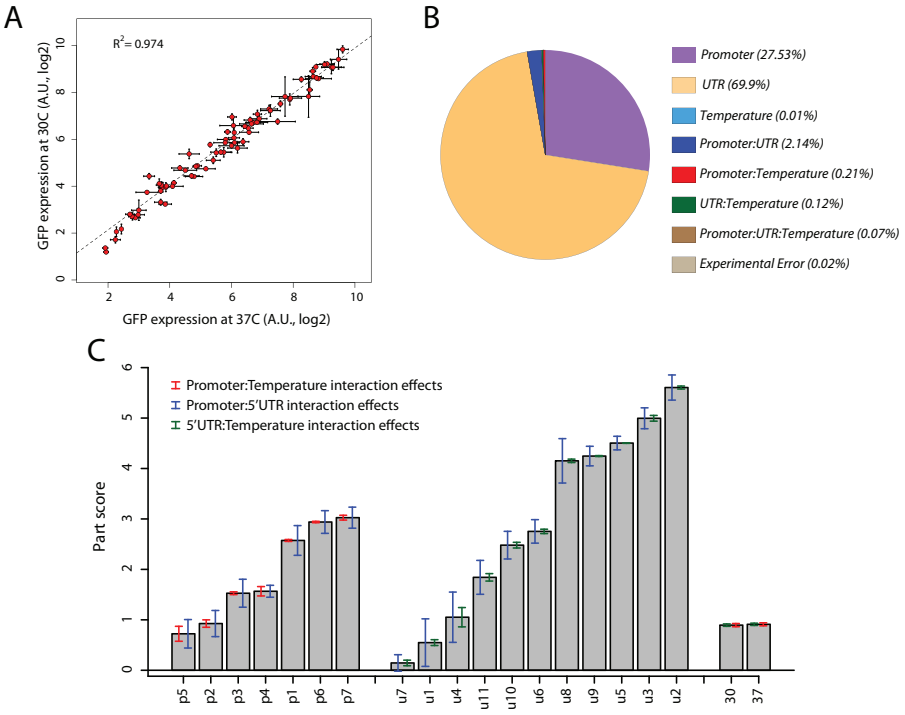


Figure 3.6 Genetic elements performance across two temperatures.
(A) Fluorescence measured at 30 °C and 37 °C are highly correlated. (B) ANOVA indicates that temperature effect on overall expression is very weak (0.01%). We also observed that temperature has a greater impact on Promoters (0.21%) than on 5'UTRs (0.12%). (C) Estimated part scores and deviations across different temperature conditions (30 °C and 37 °C). Parts p5 and u4 show higher sensitivity to temperature change.

3.4.7 Predicting part performance given limited measurement resources

We determined if and how our framework might be used to best estimate the expected average performance of new parts without having to construct and test all new possible part combinations. If there were no interactions among elements, then measurement of any new part within just a single combination of elements would be sufficient to perfectly estimate its performance across all elements. Alternatively, if each combination of elements produces highly specific effects then all combinations might need to be assembled and tested.

Using the genetic elements studied here as a test set, we first observed how the quality of prediction for the expected average activity of a Promoter increases as the number of 5'UTRs with which it is tested increases (Materials and methods). For example, estimating the quality of the most context sensitive Promoter, p1, using any single p1:5'UTR combination produces a ~8-fold range in estimated p1 activity (Figure 3.7A). As the number of 5'UTR combinations used to estimate p1 activity increases the accuracy of the p1 expected average activity improves. Similar trends were observed for other promoters, and also when one or more Promoters were used to estimate the activity of a 5'UTR (Figure 3.7B). Taken alone these results might suggest that any new part must be tested with all other possible combinations of parts to estimate its expected average activity. Such work would quickly become prohibitively expensive as the number of parts and resulting part-part combinations increases. However, we noted that a few parts consistently produced relatively accurate estimates of ensemble-average part activities even when just a single part combination was tested (Figure 3.7A-B, N = 1 columns).

To better understand these observations we computed the aggregate error in estimating the activities of all Promoters (Figure 3.7C) and 5'UTRs (Figure 3.7D) using varying numbers and combinations of 5'UTRs and Promoters, respectively (Materials and methods). We found empirically that a limited number of measurements could be used to systematically estimate ensemble-wide part activities of Promoters and 5'UTRs with reasonable accuracy. For example, the activity of any Promoter could be estimated to within 15% of its average activity across all 5'UTRs by using just two 5'UTR measurements (u11 and u6). Similarly, the activi-

ty of any 5'UTR could be determined to within 15% accuracy using just two Promoter measurements (p5 and p6). Thus, for at least some biological element types, once a full combinatorial mapping is established for a particular interaction or context variable, a greatly reduced number of experimental tests were sufficient to accurately estimate the expected ensemble-average activity of new parts. Stated differently, following an initial seeding via brute force combinatorial measurements, increasingly more efficient, affordable and accurate part characterization can be realized via centralized facilities or distributed efforts (below).

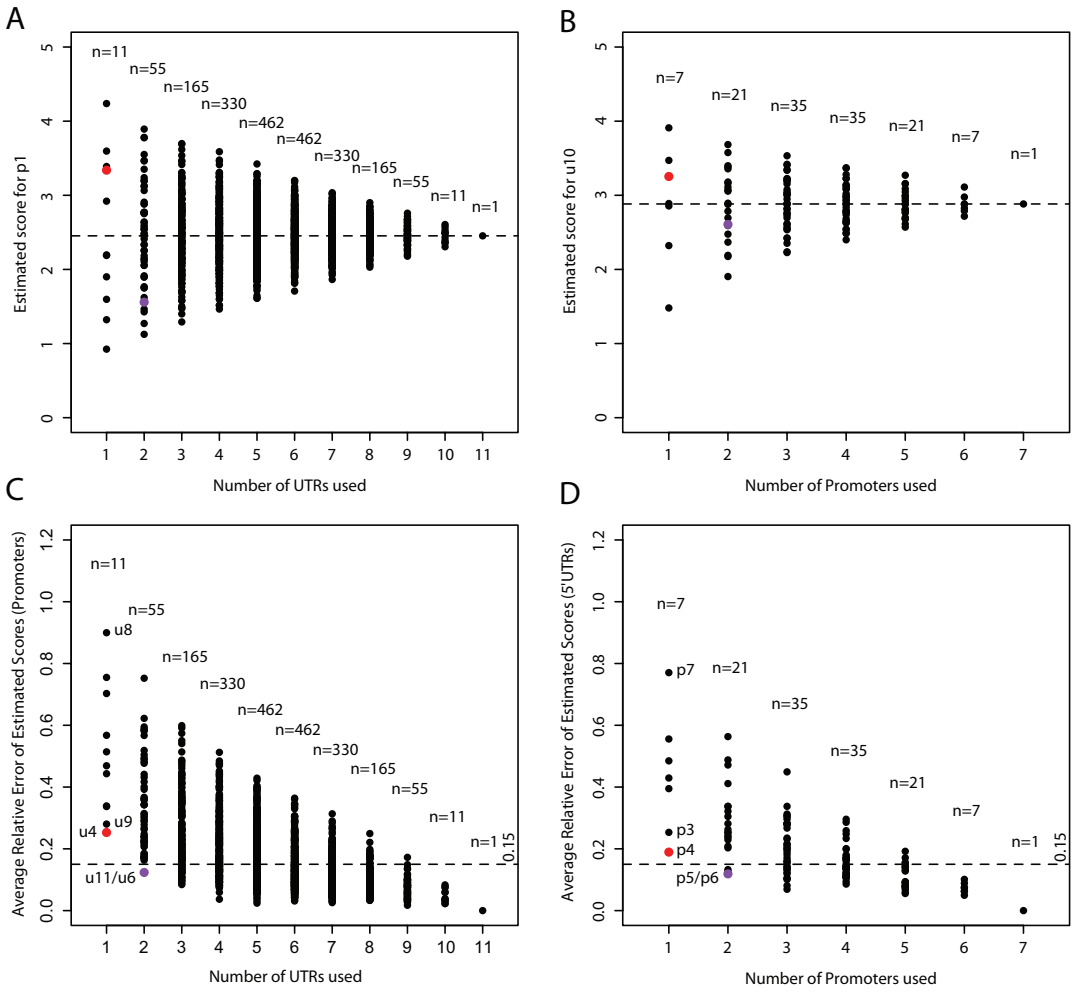


Figure 3.7 Estimation of part activity with limited measurements
(A) Estimated activity for the Promoter 'p1' with increasing number of 5'UTRs. The number (n) of possible unique 5'UTR combinations as a function of the number of 5'UTRs tested as noted. (B) Estimated activity of the 5'UTR 'u10' with increasing numbers of Promoters. (C) The relative error, averaged across all Promoters, in estimating the activities of Promoters with increasing numbers of 5'UTRs. (D) The relative error, average across all 5'UTRs, in estimating the activities of 5'UTRs with increasing numbers of Promoters. The individual part (red) and part pair (purple) that give the highest accuracy in estimating the activity of any new element are colored.

3.5 Discussion

We developed a simple linear ANOVA-based method to quantify the average activity scores of genetic elements and also variation in activity across changing contexts. We applied the method to estimate the activities of transcription and translation control elements widely used in gene expression engineering within *E coli*. By carefully controlling external and culture context variables we were able to isolate and quantify subtle functional couplings among elements that arise as elements are reused in combination (Figure 3.4). We found that interactions between 5'UTRs and two widely used fluorescent reporters produce the largest interaction factor impacting gene expression (Figure 3.3). Of note, the rank ordering of 5'UTR activities differ between the two genes used here. In contrast, the rank ordering of Promoters was the same across the different 5'UTRs and GOIs tested here. Hence, the expected performance of promoters is likely more consistent across applications. Finally, we demonstrate that our framework also provides a basis for choosing a minimal set of best elements against which to estimate the activities of new parts (Figure 3.7), thereby providing a method to improve the efficiency of efforts focused on the systematic characterization and production of self-consistent libraries of standard biological parts (Endy, 2005). The main idea is that a generic and relatively small set of parts/factors can be initially tested in multiple combinations to find the general rules governing their interactions. If minimal interactions are found, the characterization of new parts, functionally homogeneous to the initial test, can be achieved with minimal experimental labor.

Our results demonstrate the ability to systematically quantify and prioritize subtle functional couplings arising among genetic elements as elements are reused in combination (Figure 3.3). While such effects have been observed in isolation previously, several advantages accrue from more systematic analysis spanning multiple element and junction types. For example, recent work to regularize the 5' termini of mRNA using ribozymes or a CRISPR (clustered regularly interspaced short palindromic repeats) associated protein to cleave upstream sequences contributed by irregular promoters considered "promoter-gene" couplings as an aggregate junction (Lou et al., 2012; Qi et al., 2012). Splitting this bulk junction into all its encoded elements (i.e., transcription, translation, and gene of interest) might have instead

allowed simple regularization of the +1 sequence encoded from the selected promoters. Stated differently, careful attention could first ensure that irregular physical boundaries of elements selected from existing sources do not needlessly create complicating couplings across functional part boundaries. From such a foundation, many approaches could then be tested to overcome remaining functional couplings spanning more complicated element-element junctions. For example, following earlier observations (de Smit and van Duin, 1990) and confirmed by our analysis here, we determined that mRNA secondary structures at the 5'UTR:GOI junction are correlated with variation of 5'UTR scores, and used this observation to motivate the future implementation of an architecture for translation initiation elements whose activities are insensitive to changes in the coding sequence of downstream genes (see Chapter 4).

We note that our introduction of a secondary performance score, "quality," to define and track how genetic element activities vary across changing contexts can lead to paradoxical, at first, labeling. For example, we would score a "promoter" element that never initiates transcription in any and all contexts as a "high quality promoter encoding zero activity," whereas an element that in some contexts initiates transcription and in others fails would be a "low quality promoter encoding intermediate activity." Genetic element "quality" in these two examples captures to what extent users of elements can rely on the reported behaviors. For example, a "promoter" that was known to never initiate transcription would be of particular value in establishing negative controls used in quantifying both transcription promoters and terminators.

In conclusion, we caution that a simple linear model combined with ANOVA will likely not perfectly or, eventually, even usefully represent the activity and quality of genetic elements as increasing numbers of element types and environmental factors are considered simultaneously. The method also does not track cell-cell variation in activities, resource utilization, and many other important factors. Nevertheless, we have established that at least four key variables and their associated interactions can be observed, with subtle quantitative interactions impacting genetic element activity quantified given careful control of extrinsic physical variables. Extending even a simplistic approach to additional variables, genetic or environmental

(Figure 3.6) that results in the lowering of part quality scores will help to prioritize future work to improve parts.

Chapter 4

Gene Expression Control via Standard Transcription and Translation Initiation Elements

This chapter is based on the article:

Vivek K Mutalik, **Joao C Guimaraes**, Guillaume Cambray, Colin Lam, Marc Juul Christoffersen, Quynh-Anh Mai, Andrew B Tran, Morgan Paull, Jay Keasling, Adam P Arkin and Drew Endy. (2013). Precise and reliable gene expression via standard transcription and translation initiation elements. *Nature Methods*, *in press*.

4.1 Abstract

The rational engineering of biological systems is limited by our inability to reliably predict the quantitative behavior of novel combinations of genetic elements. Here, we develop an expression cassette architecture that defines boundaries and junctions for genetic elements controlling transcription and translation initiation in prokaryotes: transcription elements encode a common expected mRNA start and translation elements use an overlapping genetic motif found in many natural systems. We engineered libraries of constitutive promoters along with translation initiation elements following these definitions. We measured activity distributions for each library and selected elements that collectively encode expression levels across a 1,000-fold observed dynamic range. We made and measured all combinations of curated elements, demonstrating that arbitrary sequence distinct genes can be reliably expressed to within twofold relative target expression windows with ~93% reliability. We expect the element definitions validated here can be collectively expanded and further improved to define public domain parts collections that enable reliable forward engineering of gene expression at genome scales.

4.2 Introduction

One main goal of synthetic biology research is to make the engineering of biology easier (Endy, 2005; Purnick and Weiss, 2009). DNA synthesis and assembly has progressed to where entire metabolic pathways, chromosomes and genomes can now be synthesized and transplanted (Carr and Church, 2009; Ellis et al., 2011; Gibson et al., 2010). However, our capacities to rationally design increasingly complicated genetic systems as enabled by improving DNA construction methods has not kept pace (Lu et al., 2009; Purnick and Weiss, 2009). One of the greatest claimed barriers to efficient and scalable genetic design is the lack of standard parts that can be reused reliably in novel combinations (Endy, 2005; Keasling, 2010). Many examples instead highlight, even within well-studied organisms such as *E. coli*, how seemingly simple genetic functions behave differently when redeployed in novel settings (Cardinale and Arkin, 2012; Kittleson et al., 2012). For example, a prokaryotic ri-

bosome binding site (RBS) element that initiates translation for one coding sequence might not function at all with a second sequence distinct protein (Salis et al., 2009). If the genetic elements that encode control of central cellular processes such as transcription and translation cannot be reliably reused then there is little chance that higher-order objects encoded from such basic elements will obtain emergent reliability within larger-scale systems.

Standard biological parts could, in theory, enable hierarchical abstraction of biological functions (Canton et al., 2008; Endy, 2005; Rosenfeld et al., 2007). The behavior of integrated genetic systems could then be represented via simpler models of individual elements, and ultimately mapped to underlying genetic sequences whose encoded functions are dependent upon a limited number of measurable or calculable intrinsic variables. Such abstraction of function seems necessary to manage biological complexity and also to allow disparate researchers to contribute to the collective acceleration of capacities that enable the engineering of increasingly complicated genetic systems. However, progress to date has too-often been limited to arguments-by-analogy that invoke abstraction as realized in other fields of engineering for which the physical substrate is simpler and rules of abstraction long established (Cambray et al., 2011; Lu et al., 2009).

Here, we engineer ~500 transcription and translation initiation elements that are compatible within a standardized genetic context, or 'Expression Operating Unit' (EOU), that enables predictable forward engineering of gene expression over a wide dynamic range. We characterize representative parts for each type by testing more than 1,200 part-part combinations to establish and validate functional composition rules while quantifying scores for part activity. From these data we can also estimate the "quality" of each part, a second-order statistic that represents to what extent the activity of a part varies across changes in context (see Chapter 3). Our results demonstrate that a more physically complex design for the control of translation initiation when combined with standardized transcription control elements can lead to simply and predictively modeled gene expression in *E. coli*. Such consistency begins to make practical the quantitatively precise reuse of genetic objects encoding two core biological functions, transcription and translation initiation, that underlie all cellular biotechnology.

4.3 Materials and methods

4.3.1 Bacterial strains, plasmids and growth conditions

Detailed information on sequence designs, plasmid maps and corresponding experimental datasets for each construct developed in this study are available in a public repository database at http://biofab.org/data.

The Promoter:5'UTR and 5'UTR:GOI combinatorial libraries driving the expression of *gfp* (Pedelacq et al., 2006) and *rfp* (Campbell et al., 2002) was carried on medium copy vectors and assembled using the Golden Gate cloning method (Engler et al., 2008).

E. coli strain BW25113 was used for plasmid construction purposes and for fluorescence measurements. All strains were grown in MOPS EZ rich media (Teknova) and all the experiments were conducted in triplicate (replicate assays from independent overnight liquid cultures).

4.3.2 Expression operating unit (EOU)

The EOU is made up as a minimal unit of genetic expression (expression cassette) and an additional flanking region that may help insulate EOU parts. Minimal unit of genetic expression is made up of a promoter with defined transcription start site (pTrc*, a constitutive promoter, -35 to $+1$) (Lutz and Bujard, 1997), a translation initiation element (TIE), a protein-coding region (e.g., a *gfp* reporter), and a terminator (3'UTR, dbi terminator (Lee et al., 2011)).

To provide functional insulation to the EOU from cryptic promoters, ribosome binding site-like regions, intrinsic terminators, AT-rich up element like features, we have introduced an additional upstream region comprised of three out of frame stop codons, an intrinsic terminator (McDowell et al., 1994), a transcriptional pause site (Kireeva and Kashlev, 2009) and an insulator region (Davis et al., 2011). Here, the upstream and downstream terminators are designed and positioned to reduce the interactions between the EOU and the immediate genetic context. The EOU thus provides a standardized and well-defined context that insulates functional parts within the EOU from neighboring genetic contexts and provides a more reliable platform for characterization of parts (Figure 4.1).

4.3.3 Design of promoter and bicistronic library

For generating the promoter library, we used two distinct approaches. In the first approach, we randomized the -35 and -10 motifs of a strong pTrc* promoter to generate a randomized promoter library (RPL) composed of 200 clones. In the second approach, a modular promoter library (MPL) was created by the combinatorial assembly of three different modules — (1) up element and -35 motif; (2) spacer region; and (3) -10 and spacer from -10 to +1 — of five well characterized promoters of different strength producing a total of 125 (5^3) different clones.

For generating the randomized bicistronic design (BCD) library, we created variants of second SD (SD2) of BCD such that three nucleotides upstream and downstream of 'GGA' motif of SD2 were randomized (NNNGGANNN, Figure 4.1). About 200 colonies were picked and positive clones were confirmed by sequencing. We discarded mutants with stop codons within first cistron, in addition to deletion and insertion mutants within the leader peptide library.

4.3.4 *In vivo* assays using the plate reader and flow cytometer

Cultures were grown in 2 ml 96 deep well plates containing 400 µl of MOPS EZ rich media (Teknova, cat.# M2105) with appropriate antibiotics, inoculating 3 µl from thawed glycerol stocks. Cultures were grown overnight in 96-well plates at 37 °C with shaking at 900 rpm on Multitron shaker (Inforys-HT).

For the flow cytometer assays, overnight cultures were diluted 1:50 into a final volume of 200 µl fresh media with appropriate antibiotics in 96 deep well plates and grown for 2 hours (to exponential phase with OD_{600} in the range of 0.3 to 0.5 in microplate reader) at 37 °C with shaking at 900 rpm on Multitron shaker. Cultures were diluted 1:2,000 in chilled and filtered PBS (pH 7.4) containing 500ug/ml streptomycin in chilled 96-well clear plates (Costar) and immediately subjected to flow cytometer analysis. We used a Guava® EasyCyte flow cytometer (EMD Millipore, Hayward, CA) equipped with microcapillary and auto-sampling capabilities. For each sample, 2,000 cells were collected.

4.3.5 RNA structure prediction

To evaluate the potential for forming stable inhibitory structures between different chosen genes of interest (GOIs) and TIEs, we used UNAfold software (Markham and Zuker, 2008) to predict the minimum folding energy (MFE) structure conformation. We considered the junction region to be comprised between the position -26 and +37 with respect to the translation start site. These boundaries were selected based on the size of the monocistronic design (MCD) 5'UTR and the 36 nt region of GOI, respectively.

To evaluate the affinity between SD2 sequences (in BCD and in MCD) and the anti-SD region of the 16S rRNA (acctcctta), we used UNAfold software (Markham and Zuker, 2008) to calculate the hybridization energy for the resulting RNA duplex. We considered the region spanning from position -26 and -1 with respect to the translation start site for both BCD and MCD constructs. We then correlated the calculated free energy with the fluorescence measurements from fusion reporters from both MCD and BCD constructs.

4.3.6 ANOVA models

To understand the contribution and coupling between translation element (i.e., MCD and BCD) and the GOI on the overall gene expression, we used the ANOVA framework developed in Chapter 3. Two different linear models were considered to evaluate the overall effect of parts and parts-interaction for the two datasets:

$$\log_2(\text{Fluorescence}_{ij}) = \alpha + \text{TIE}_i + \text{GOI}_j + (\text{TIE:GOI})_{ij} + \varepsilon_{ijk} \quad \text{for } i = (1 \text{ to } 22);\ j = (1 \text{ to } 8); \qquad \text{Eq. 4.1}$$

$$\log_2(\text{Fluorescence}_{ijk}) = \alpha + \text{P}_i + \text{TIE}_j + \text{GOI}_k + (\text{P:TIE})_{ij} + (\text{TIE:GOI})_{ik} + (\text{P:GOI})_{jk} + (\text{P:TIE:GOI})_{ijk} + \varepsilon_{ijk} \quad \text{for } i = (1 \text{ to } 14);\ j = (1 \text{ to } 22);\ k=(1,2); \qquad \text{Eq. 4.2}$$

where Fluorescence_{ij} is the fluorescent output signals measured from a genetic construct comprising a translation initiation element, TIE_i, and a gene of interest, GOI_j. $\text{Fluorescence}_{ijk}$ is the fluorescent output signal measured from a genetic construct comprising a transcriptional element *i*, a translation element *j* and a gene *k*. P:TIE, TIE:GOI, P:GOI and P:TIE:GOI denote the two- and three-way parts-

interactions. α is the overall average signal, and the term ε_{ijk} represents the error term for each particular construct.

Given the high degree of independence of the parts developed here, we hypothesized that a regression model for predicting expression from the identity of a particular Promoter and BCD trained on expression measurements of a given reporter could be used to predict the expression of other GOIs. To do this we considered Eq. 4.2 and used GFP-only measurement to estimate the regression coefficients for the different Promoters and BCD variants. This model was then used to predict the performance of both Promoter:BCD:RFP and BCD:GOIs libraries.

4.3.7 Expression probability calculations

The probability of observed expression level falling within a factor two of the predicted expression level was determined using the following method: for each of the strains, the mean-normalized $\log_2$ fluorescence values (observed) and the predicted values (using the model described above) were calculated for a total of 924 pairs of observed and predicted values. The absolute difference between the observed and predicted values was calculated, and the percent of absolute difference values being less than 1 was empirically determined to be 92.7%.

4.4 Results

4.4.1 Prioritizing parts puzzles

In the study presented in Chapter 3, we systematically assembled and tested all combinations of frequently used prokaryotic transcription and translation control elements to quantify average part activities and also variation in activities as parts are reused in novel combinations. In particular, we quantified how unknown interactions arising from irregular part-part junctions account for ~17% of the total variation when "irregular" transcription and translation elements are reused in combination; "irregular" refers to genetic elements whose boundaries have not been determined or refined in support of reliable functional composition. Thus, we focused here on developing rules that could define a genetic layout architecture for gene expression cassettes that eliminate functional uncertainty arising from the reuse of

transcription and translation initiation elements with any sequence-distinct gene of interest (Figure 4.1). While we herein only consider three elements — transcription initiation element (or Promoters), translation initiation element (TIE, usually a 5'UTR), and Genes Of Interest (GOI) — and two adjacent element-element junctions — Promoters:TIEs & TIEs:GOIs — subsequent work can expand the EOU architecture and variants thereof in a distributed and asynchronous fashion.

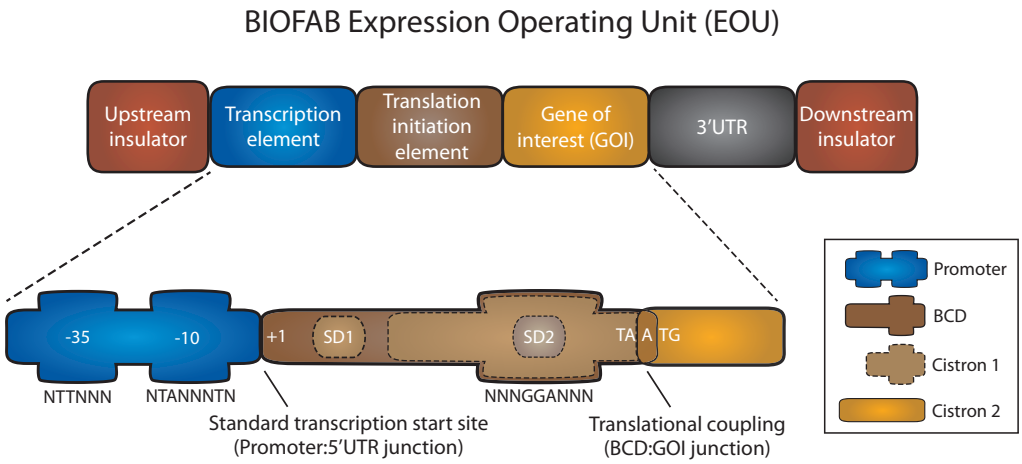


Figure 4.1 Expression operating unit (EOU)
An EOU defines the boundaries and junctions of functional genetic elements underlying the expression of genes. In this work we established boundaries and junctions for transcription elements, translation initiation elements, and genes of interest. The element type and the standard junctions between elements that best enable reliable reuse of elements in novel combinations are detailed. The bicistronic translation initiation element has two Shine-Dalgarno (SD) motifs and a small cistron region (main text). Variable regions used to produce different part variants are represented by the wider icons.

Recent studies have focused on regularizing a Promoter:5'UTR junctions via active enzymatic processing of mRNA (Lou et al., 2012; Qi et al., 2012). However, our study of many Promoter:5'UTR and 5'UTR:GOI combinations found that variation in translation initiation rates arising from irregular 5'UTR:GOI junctions produced most of observed expression irregularities (14 of 17% total, see Chapter 3). Given this information and further noting that, in prokaryotes, irregularities arising specifically across 5'UTR:GOI junctions cannot be eliminated by enzymatic cleavage between a Shine-Dalgarno (SD) sequence and translation start codon, we decided to first pursue enabling the reliable initiation of translation for any gene coding sequence.

Differential formation of mRNA secondary structures spanning 5'UTR:GOI junctions that then influence ribosome binding or initiation has long been recognized as a major determinant of variation in translation initiation rates (Hall et al., 1982). For example, one study concludes "that reusing the same well-characterized RBS (ribosome binding site) sequence for different proteins — a common practice — is not likely to work reliably" (Salis et al., 2009). Instead, current methods require the construction of multiple RBS variants followed by experimental screening to obtain desired expression levels, presumably through changes in translation initiation efficiency (Keasling, 2010; Salis et al., 2009). Alternatively, the best available computational tool for designing context optimized translation control elements for use in *E. coli* gives a ~47% chance to design elements that express proteins to within twofold of a target expression level (Salis et al., 2009). We note that such quantitative precision in detailing the compositional reliability of designer genetic elements is rare yet necessary to evaluate and improve current engineering practice. However, given current forward engineering design capacities, if a specific protein expression level is required then repeated sequence-distinct design attempts must be synthesized and tested experimentally, often resulting in combinatorial increases in required design attempts as system complexity (i.e., number of genes) increases.

We instead sought an architecture for TIE:GOI junctions that would allow for a TIE to more reliably encode a distinct and sequence-specific translation initiation rate without sensitivity to variation in the coding sequence of downstream GOI. We reconsidered past work with difficult-to-express proteins and also reexamined the detailed architecture of natural polycistronic operons (Das and Yanofsky, 1984; Makoff and Smallwood, 1990; Mendez-Perez et al., 2012; Oppenheim and Yanofsky, 1980; Schoner et al., 1986; Schumperli et al., 1982; Spanjaard and van Duin, 1989). Of particular interest were past examples in which a second independently translated coding sequence is positioned immediately upstream of, or slightly overlapping with, the coding sequence of any given GOI (Makoff and Smallwood, 1990; Schoner et al., 1986). In such arrangements the RBS for the GOI is embedded within the coding sequence of the upstream gene and, thus, translation of the downstream cistron may be coupled to translation of the upstream cistron. More specifically, the intrinsic helicase activity of ribosomes arriving at the stop codon of an upstream

cistron might eliminate inhibitory RNA structures that would otherwise disrupt translation initiation of the downstream GOI (Qu et al., 2011; Takyar et al., 2005).

To explore if overlapping genetic elements and active translation coupling might improve translation initiation reliability, we considered reported genetic designs that encode short leader peptides followed by a downstream GOI (Makoff and Smallwood, 1990; Schoner et al., 1986). One of the designs (Makoff and Smallwood, 1990) encodes a 16 amino acid leader peptide within a first cistron wherein the stop codon overlaps by one base pair with the start codon of the downstream gene (TTATG) (Figure 4.1). The leader peptide is synthesized by ribosomes that bind to an upstream SD core sequence (SD1); translation of the downstream GOI is thought to result, primarily, from SD1-directed ribosomes that recognize and re-initiate translation via a second SD site (SD2) that is encoded entirely within the coding sequence of the leader peptide (Das and Yanofsky, 1984; Makoff and Smallwood, 1990; Oppenheim and Yanofsky, 1980; Spanjaard and van Duin, 1989). We termed this translational coupling architecture a "bicistronic design" (BCD) to acknowledge the major difference from conventional "monocistronic designs" (MCD) in which translation of coding sequences initiates from a SD site that does not overlap with other functional sequences. We reconfirmed that, unlike SD motifs encoded within MCDs, those encoded within BCDs could initiate protein synthesis even if the coding sequence for the GOI contains a perfect reverse complement to the cognate SD site (Figure 4.2), implying that translation from SD1 disrupts mRNA structure spanning the junction between cistrons such that translation initiation from SD2 is restored.

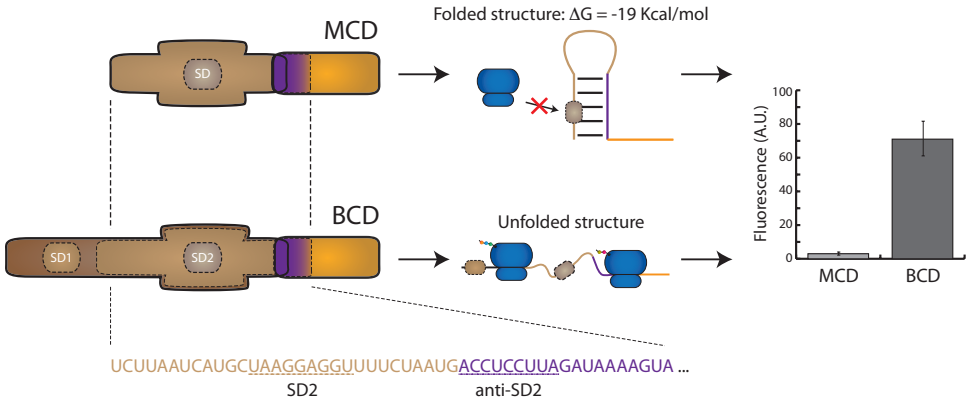


Figure 4.2 Translational coupling overcomes strong RNA structures
The same promoter pTrc* and translational initiation element context was used to express one GOI (in purple) fused in-frame to a green fluorescent protein (GFP) reporter (in orange). The GOI coding sequence contains a perfect reverse complement sequence to the core of SD (SD and SD2, in the MCD and BCD, respectively; underlined sequences) and is expected to form a strong stem loop. In the MCD this structure is formed and SD is sequestered, resulting in very low probability of binding between the ribosome and the mRNA (top). In the BCD, SD2 is entirely embedded within a short leader cistron that is translated by ribosomes binding to SD1 and, presumably, disrupt potential RNA secondary structure formation at the SD2:GOI junction (bottom). The experimental data indicates that BCD can express well the GOI, whereas the MCD only synthesizes a very low amount of protein.

4.4.2 Precise and reliable translation initiation

We then sought to establish if the BCD could be generalized so as to initiate synthesis of many sequence distinct proteins across a wide range of translation initiation rates. Different translation initiation rates can be realized by varying the SD sequence to generate differential ribosome binding affinities (Barrick et al., 1994). Though the significance of specific SD2 sequence elements has been recognized in a few naturally coupled cistrons (Das and Yanofsky, 1984; Oppenheim and Yanofsky, 1980; Schumperli et al., 1982), there are no reports of engineering a library of SD2 variants to fine tune expression of a downstream GOI. We hypothesized that, for a given SD1 sequence element, a wide range of translation initiation rates could be realized within a BCD by varying the embedded SD2 sequence (Figure 4.1). We randomized a SD2 motif preserving a three nucleotide consensus core (NNNGGANNN) and obtained a couple hundred sequence distinct clones across a ~600-fold range of expressed reporter protein levels (Figure 4.3A). We also found that the observed fluorescence was significantly correlated with hybridization energy between the 16S rRNA and the SD2 sequence ($r = 0.6$, P-value < 0.001; Figure

4.3B). This observation suggests that the strength of translation initiation for the GOI is partially determined by the affinity between the ribosome and the SD2 sequence.

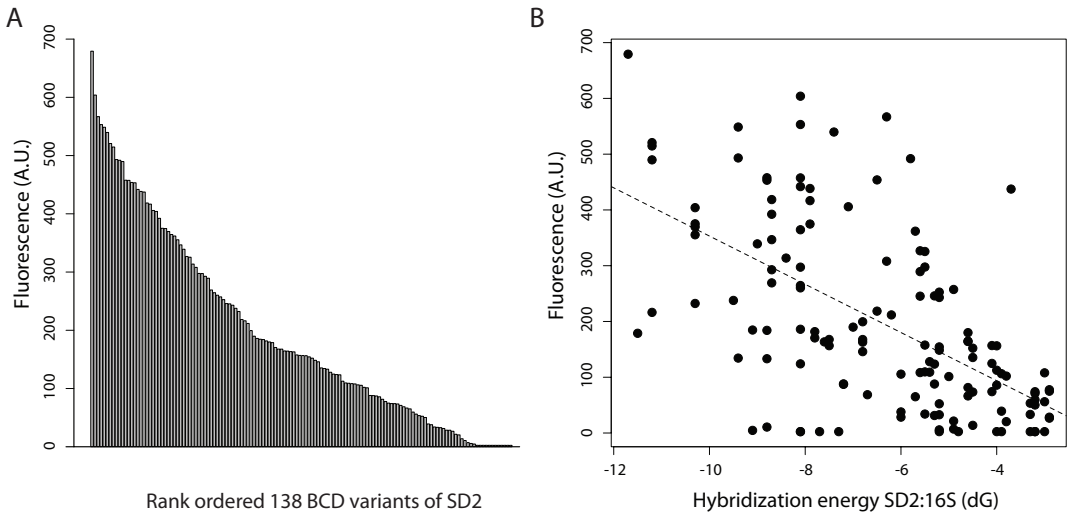


Figure 4.3 Generation of BCD variants of SD2
(A) The SD2 region within the BCD architecture was randomized and variants were characterized by measuring GFP fluorescence. Rank ordered BCD (SD2) variants span across a ~600-fold range of expression. (B) Predicted hybridization free energy between 16S rRNA and core SD2 sequences are well correlated ($r = 0.6$, P-value < 0.001).

From this BCD library, we chose 22 different strength SD2 candidates to test if each retained its relative encoded initiation strength when used to express sequence distinct genes. Also, to directly compare the performance of BCDs to conventional MCDs, we used the same SD2 sequences within MCDs. We then assembled a test panel of 14 sequence-distinct reporter GOIs by fusing the first 36 nucleotides from the coding sequences of popular transcription factors and important enzymes in-frame to the second codon of green (*gfp*) or red (*rfp*) fluorescent proteins genes (Figure 4.2). We selected a 36 nucleotide GOI leader since this coding sequence length is thought sufficient to encompass effects of ribosome footprint and mRNA secondary structure formation on translation initiation (Kudla et al., 2009; Steitz, 1969; Yusupova et al., 2001). As additional GOI controls, we included a chimeric reporter protein encoding a full-length *tetR* coding sequence fused in-frame to *gfp*, as well as the full length sequences of *gfp* and *rfp* reporters. RNA free energy predictions indicated that our GOI set is expected to form a wide range of stable

mRNA secondary structures spanning sequence distinct BCD:GOI junctions (-24 < ΔG < -7 kcal/mol).

We assembled two full combinatorial test libraries in which 22 MCDs or 22 BCDs were used to translate the 14 chimeric reporter GOIs. Then, we quantified absolute expression levels by measuring single cell fluorescence from all 308 MCD:GOI and 308 BCD:GOI combinations (Figure 4.4). We observed, as expected, that the synthesis of proteins from conventional MCDs is highly sensitive to changes in the coding sequences of genes (Figure 4.4A, ~0.43 average Spearman rank correlation (*rho*) between any two GOIs). Conversely, we then observed that the same 22 SD2 motifs, when used within BCDs, maintained relative fluorescence levels regardless of the downstream GOI (Figure 4.4B, ~0.85 average *rho* between any two GOIs).

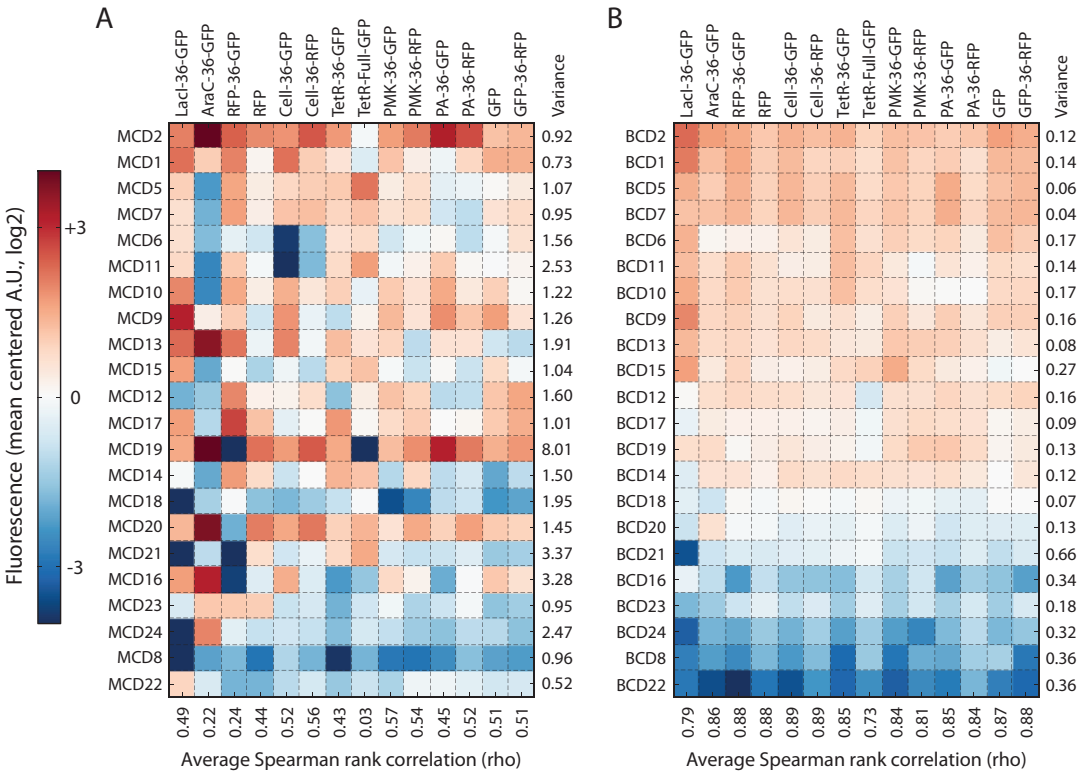


Figure 4.4 Combinatorial library of MCDs/BCDs and GOIs
(A) Gene expression of the 22 different strength MCDs (rows) and 14 sequence-distinct GOIs (columns). (B) The same SD2 sequences used in the MCD but encoded within BCD. Rank ordering for the heatmap was established via data of (B). The 14 GOIs are encoded via the first 36 nt N-terminal coding sequences of the proteins: LacI, AraC, RFP, Cellulase (Cell), TetR, PhosphoMevalonate Kinase (PMK), and Penicillin Acylase (PA). For each heatmap, variance in mean centered $\log_2$ expression levels per GOI (right), and average *rho* (bottom) between expression levels of each GOI and all the other GOIs.

We then applied the ANOVA framework described in Chapter 3 to quantify intrinsic part performance, as well as the part-part performance variability arising from the composition between the different parts (Figure 4.5 and Table 8, Materials and methods).

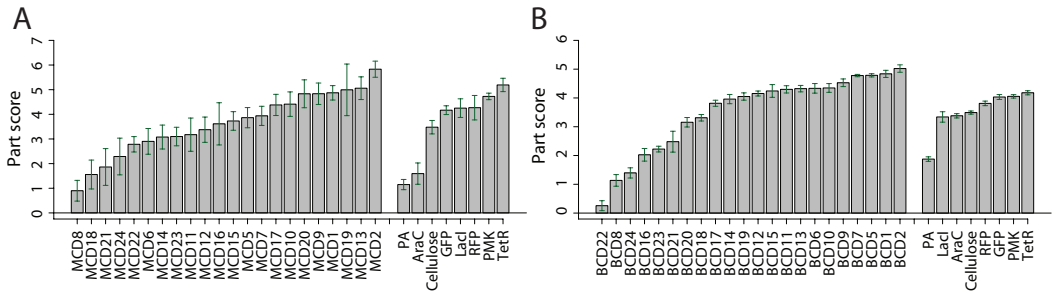


Figure 4.5 Performance and quality scores for MCD and BCD elements.
Element scores (bar heights) represent the primary activity of each TIE variant (MCD, panel A and BCD, panel B) and coding sequence (GOI_j) to observed fluorescence (Materials and methods). Error bars indicate the standard error of all interactions involving each MCD/BCD element and GOI. As such error bars reflect the reliability of element performance relative to changes in proximal genetic context.

Table 8 ANOVA table for MCD and BCD
Sum of squares represent the actual explanation of variation in the output measurement for MCD (A) and BCD (B). The mean squares represent the average contribution of each of the factors/interactions taking into account their degrees of freedom (df). Output variables were mean centered per GOI to control for the signal bias between the two reporters (column mean sq. given GOI).

(A)	df	Sum sq.	Mean sq.	Mean sq. given GOI	%
MCD	21	792.08	38.04	38.04	83.97
GOI	7	990.99	141.57	0.03	0.0
MCD:GOI	147	1059.99	7.21	7.21	15.92
Residuals	352	6.92	0.02	0.02	0.04

(B)	df	Sum sq.	Mean sq.	Mean sq. given GOI	%
BCD	21	919.15	43.77	43.77	98.43
GOI	7	252.49	36.07	0	0.0
BCD:GOI	147	96.64	0.657	0.66	1.48
Residuals	352	12.46	0.04	0.04	0.09

Overall, the BCDs reduced variation in gene expression levels arising from irregularities spanning TIE:GOI junctions from 16% to 1.5% of the total dynamic range for gene expression (Figure 4.6A-B, Materials and methods). These improvements were realized by systematically increasing protein synthesis for TIE:GOI junctions

that encoded below average synthesis levels within a MCD context, and decreasing protein synthesis for TIE:GOI junctions encoding above average levels within a MCD context (Figure 4.6C). We determined that an equilibrium thermodynamic model based solely on the predicted free energies of binding between 16S rRNA and SD2 sequences is well correlated with observed BCD-mediated protein synthesis (BCD average r ~ -0.76 versus MCD average r ~ -0.45, Figure 4.6D), further suggesting that the BCD isolates translation initiation activity from variation in downstream gene context. Composite free energy calculations from a statistical thermodynamic model (Salis et al., 2009) — that considers binding of the 16S rRNA to the SD, mRNA base pairing and other sequence features — were less well correlated for BCDs but better correlated for MCDs (average r ~ -0.6 for both BCD and MCD-directed protein synthesis), indicating that the encoded activities of different strength BCDs are best mapped to a relatively simpler core SD2 sequence motif.

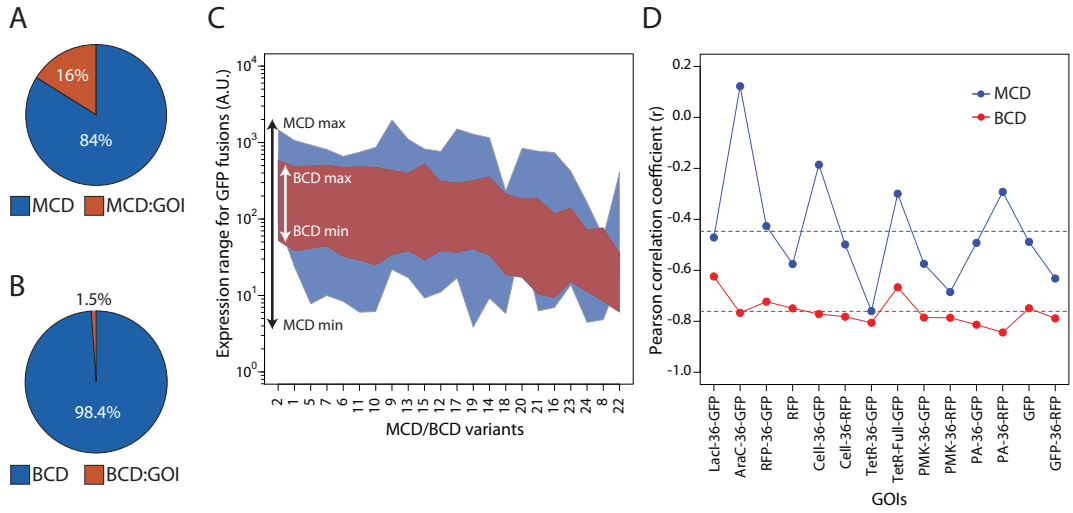


Figure 4.6 BCD encodes reliable translation initiation strengths across multiple GOIs
ANOVA in total protein synthesis realized using the (A) MCDs or (B) BCDs. (C) Comparison of absolute GFP synthesis ranges produced using MCDs or BCDs across all tested GOIs. (D) Predicted hybridization free energies between 16S rRNA and SD sequences are better correlated to expression levels for BCDs than MCDs (dashed line indicates the average correlation across all GOIs).

We also explored if BCD performance is limited to particular transcription systems or a specific internal sequences. First, we used a consensus bacteriophage T7 promoter and polymerase to transcribe BCDs and GOIs. T7 RNA polymerase synthesizes mRNA at a rate up to ~8-fold faster than native *E. coli* transcription, and thus likely results in ribosome-free 5' mRNA prior to ribosome loading and transla-

tion initiation (Iost et al., 1992), potentially leading to changes in mRNA folding or processing. We found that the T7 expression system increased average expression levels ~4-fold, as expected, and the activities of BCDs remained well correlated to those obtained with a medium strength *E. coli* promoter (*rho* = 0.93, Figure 4.7A).

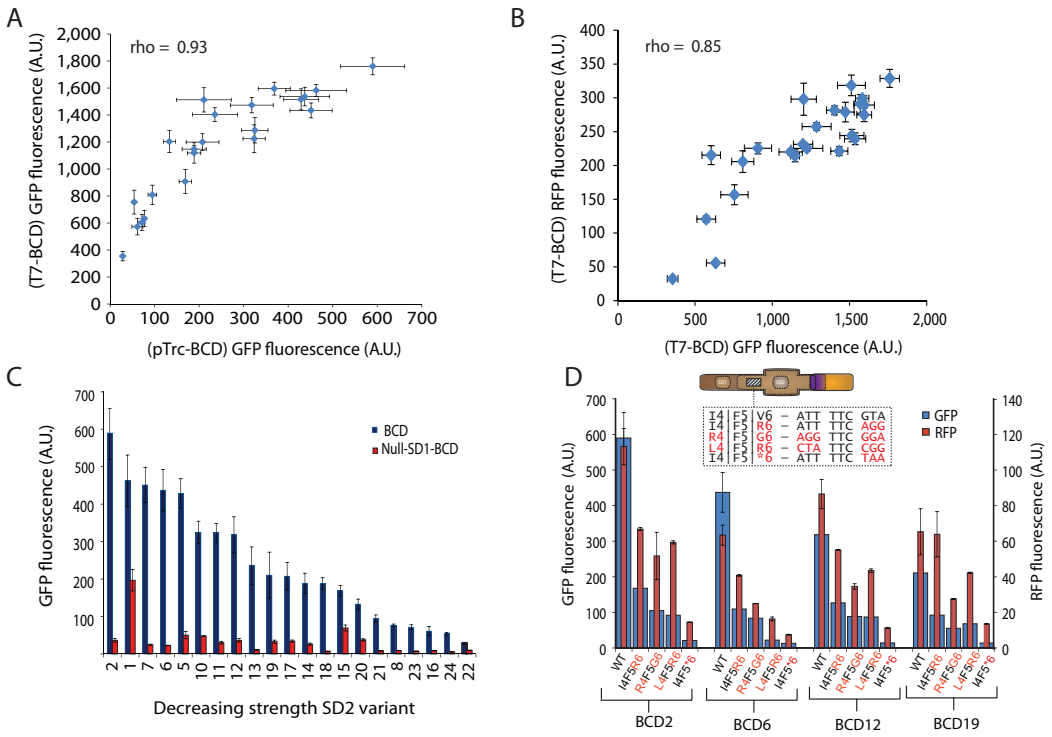


Figure 4.7 BCD retains functional reliability with alternate transcription systems and different leader cistrons

(A) Correlated gene expression levels using BCDs with an *E. coli* pTrc* promoter (x axis) or bacteriophage T7 (y axis) promoter and RNA polymerase. (B) Correlation of gene expression levels using a phage T7 transcription system with two sequence distinct GOIs. (C) Rank ordered GFP expression levels for BCDs compared to those in which SD1 is disrupted (Null-SD1-BCD). (D) Correlated expression levels using an *E. coli* promoter and the BCD's leader cistron with stop or rare codons (inset schematic) and across different strength SD2 elements (x axis, clustered groupings). Error bars indicate the standard deviation across 3 replicates.

We also confirmed that the T7 transcription system did not significantly disrupt the reliability of BCDs across changing GOI contexts (*rho* = 0.85, Figure 4.7B). Second, we demonstrated that an active SD1 motif is required to enable reliable initiation at different strength SD2 motifs and translate downstream GOIs (Figure 4.7C). Such results are in agreement with earlier studies on naturally coupled cistrons in which the significance of varying SD1 has been explored to a limited extent within the context of a stronger and unchanging SD2 sequence (Das and Yanofsky,

1984; Spanjaard and van Duin, 1989). Third, we determined that the introduction of rare codons within the leader cistron of the BCD consistently reduced expression levels without major disruption of BCD functionality, whereas adding a stop codon to the leader cistron nearly eliminated expression (Figure 4.7D).

4.4.3 Functional composition and reliable gene expression

The successful initial standardization of translation initiation elements supporting the reliable functional coupling across BCD:GOI junctions then allowed us to pursue and realize regularized coupling of transcription and translation initiation elements. Building from prior promoter engineering projects (Alper et al., 2005; Cox et al., 2007) and transcription initiation studies (Hook-Barnard and Hinton, 2007), we chose to simply regularize Promoter:BCD junctions by using promoters that encode a common +1 mRNA start, thereby hoping to avoid complicating requirements such as post-transcriptional mRNA processing (Lou et al., 2012; Qi et al., 2012). We developed a variable strength constitutive promoter library with consistent putative mRNA start sites that collectively encoded a ~900-fold dynamic range of expressed reporter levels (Figure 4.8 and Materials and methods).

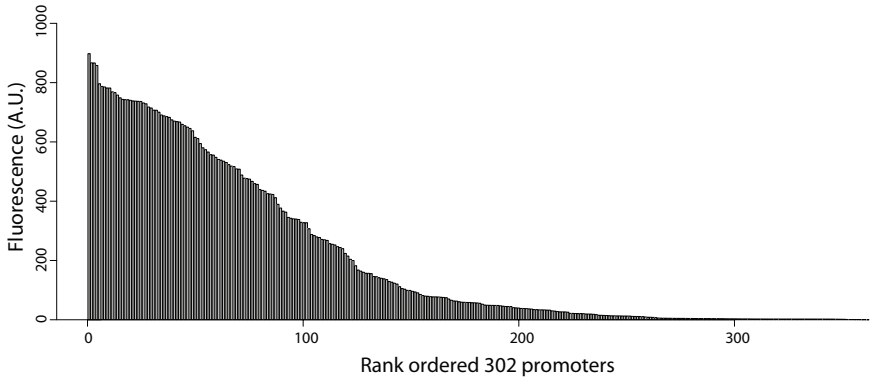


Figure 4.8 Promoter library
A rank ordered library of constitutive promoters that encode an expected common +1 mRNA (Materials and methods).

We selected 14 sequence and activity distinct promoters for further study. We assembled each promoter with all 22 BCDs, and tested expression using two sequence distinct GOIs (G/RFP, Figure 4.9A). The ANOVA of observed fluorescence levels

indicated that 98% of the total dynamic protein expression range was due to encoded differences in the intrinsic activities of individual promoters and BCDs, and not to unknown effects arising from the reuse of these expression control elements in novel combinations (Figure 4.10A and Materials and methods). Thus, the individual rank orderings for promoters and BCDs and resulting G/RFP expression levels were systematically maintained and well correlated across a 1,000-fold range for observed protein fluorescence ($R^2 = 0.85$, Figure 4.9B-C and Figure 4.10B). Moreover, a quantitative model for gene expression based only on observed GFP fluorescence levels allowed us to predict observed fluorescence for RFP and all other GOIs (Figure 4.10C and Materials and methods).

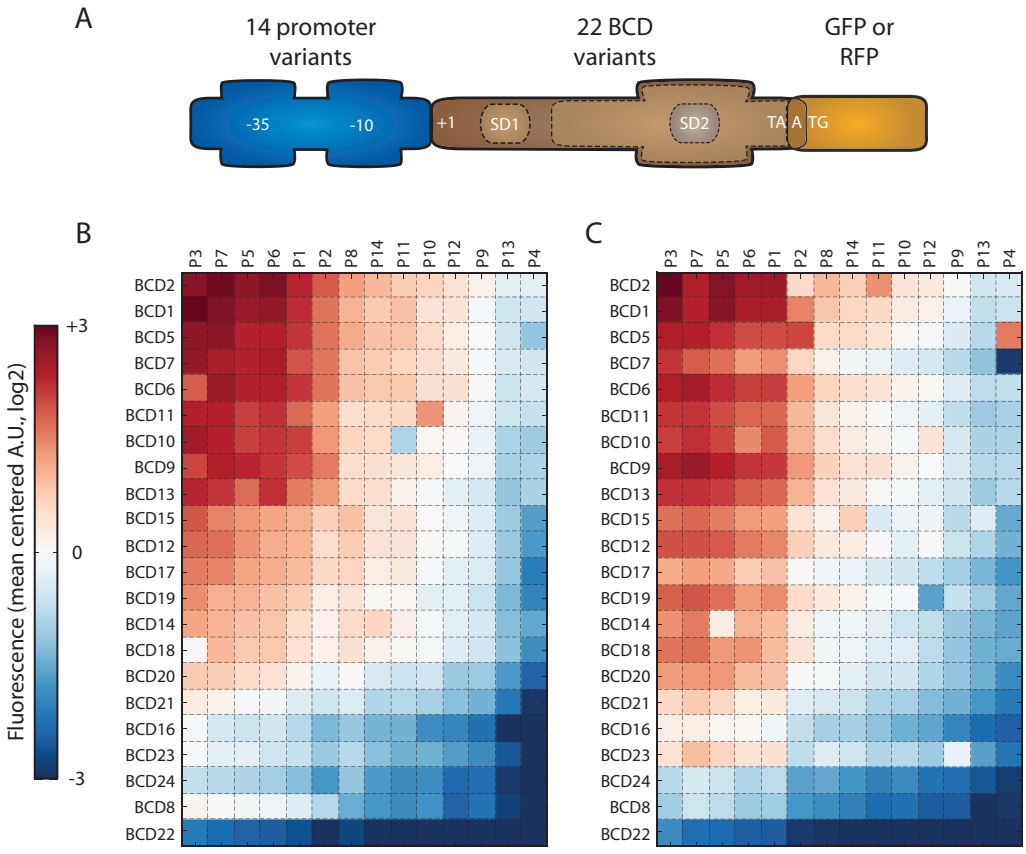


Figure 4.9 Combinatorial library of Promoters and BCDs
(A) Standard promoters produce mRNA from a common +1 nucleotide position. Translation initiation of any so-produced mRNA is entirely encoded by a separate and independent BCD. (B-C) Mean centered $\log_2$ expression levels for green (B) and red (C) fluorescent proteins via a full combinatorial library of standardized promoters (14) and BCDs (22).

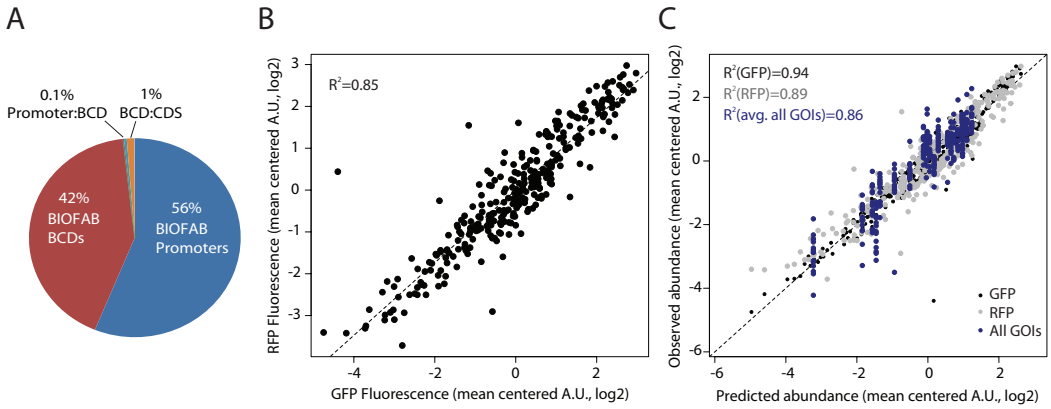


Figure 4.10 Precise and reliable gene expression via standard transcriptional and translational initiation elements

(A) ANOVA for mean-normalized expression levels from the standard promoter and BCD combinatorial library, with element- and junction-specific contributions to total expression levels as noted (Materials and methods). (B) Direct correlation of expression levels for the two different GOIs using reliable elements (R^2 = 0.85; compare with that of the use of irregular elements reported in Figure 3.2H — R^2 = 0.38). (C) Correlation of observed versus predicted protein expression levels for the Promoter:BCD:GFP library (GFP — training set), the Promoter:BCD:RFP library (RFP) and the BCD:GOI library (All GOIs) (Materials and methods).

4.5 Discussion

Our EOU architecture demonstrates how focused efforts to specifically eliminate functional complexity arising from mRNA structure effects on transcription and translation initiation can produce standard expression control elements that can be reliably reused in combination and with sequence distinct genes. So-produced genetic elements enable quantitatively precise initiation of gene expression across a 1,000-fold range in observed protein levels. Practically, users of these genetic elements should realize a ~93% chance to obtain expected GOI-normalized relative expression for a given gene to within twofold of a target level (Materials and methods). These results collectively illustrate that it is not impossible to overcome many of the challenges thought to limit the engineering of synthetic biological systems via standard biological parts: (i) lack of systematic part characterization, (ii) incompatibly of performance within part collections, (iii) variable part performance across changing (genetic) contexts, and (iv) lack of precise and predictable behavior when used (Kwok, 2010). However, just as one early reliable screw thread standard did not itself enable all of mechanical engineering, much work remains in, for example,

expanding EOU architectures to incorporate and validate additional genetic functions within *E. coli* and across additional organisms.

In establishing reliable Promoter:TIE and TIE:GOI junctions we used two distinct strategies. The Promoter:TIE junction was simply regularized by ensuring that Promoters do not contribute mRNA sequence to a standardized 5'UTR sequence, thereby ensuring simple functional decoupling. However, rendering a standard and predictably-functioning TIE:GOI junction required a genetic layout in which genetic elements were nested, overlapping, and functionally coupled as is common to many natural genetic systems (microbes, phages, viruses, and some eukaryotes) (Kozak, 1999; Oppenheim and Yanofsky, 1980; Schumperli et al., 1982; Shcherbakov and Garber, 2000). In contrast, early synthetic biology "refactoring" projects have purposefully removed such complexity and continue to eliminate it in seeking physical layout simplicity and presumed functional independence for individual genetic elements (Chan et al., 2005; Temme et al., 2012). We suspect that natural genetic systems might provide further lessons for how more complicated physical couplings can encode simpler and more reliable functional composition schemes.

The BCD could likely be used in combination with other gene expression regulatory elements and designs (Mutalik et al., 2012), to engineer synthetic polycistronic expression cassettes, or to reduce library sizes in directed evolution efforts by allowing rational choice of a few sequences that would cover desired expression parameter space. Sequence distinct BCDs are available for engineering multi-gene systems if genetic instability arising from direct repeats of DNA elements would be undesirable. Though we did not observe growth defects or other deleterious phenotypes due to expression of BCD-encoded leader peptides further studies should consider potential impacts arising from their repeated over-expression. Finally, while research to understand translation initiation within MCD contexts is relatively well established (Aitken et al., 2010), direct observation of how ribosomes reinitiate translation and overcome inhibitory mRNA structures within polycistronic operons and across varying coding sequence contexts would be helpful.

DNA sequence data and functional information detailing the performance of the standard promoters and BCDs established here have been contributed to the public domain and are freely available for use (http://biofab.org/data). Potential variation

in specific sequence distinct protein levels due to mechanisms that act downstream of translation initiation (e.g., codon usage) must still be accounted for using existing tools to obtain absolute target protein concentrations (Welch et al., 2009b). Given an expected 93% reliability rate (7% failure rate) for precision expression engineering, designers of heterologous genetic systems and tool developers working to support the engineering design process, might further explore how to best practically enable the a priori quantitative specification of desired protein synthesis levels within systems encoding up to ~10 genes.

Chapter 5

D-Tailor: Automated Analysis and Design of DNA Sequences

This chapter is based on the article:

Joao C Guimaraes, Miguel Rocha, Adam P Arkin and Guillaume Cambray. D-Tailor: automated analysis and design of DNA sequences. *Under review*.

5.1 Abstract

Recent advances in DNA synthesis and cloning technologies afford high throughput implementation of artificial sequences into living cells. However, our ability to design sequences to interrogate multifactorial biological processes and further engineer biological functions is lagging behind. We developed DNA-Tailor (D-Tailor), a fully extendable software framework for multi-objective design of synthetic sequences. Specifically, D-Tailor permits the seamless integration of multiple sequence analysis tools into a Monte-Carlo algorithm that is capable to achieve systematic design of libraries of sequences over a range of rationally defined parameters. As proof of concept, we used D-Tailor to examine three different sequence determinants of translation efficiency across the *E. coli* genome. We then showed that D-Tailor provides an efficient design algorithm to generate libraries of sequences conforming to full-factorial designs of experiments. Such libraries evenly explore the defined natural feature space and should therefore enable rigorous testing of subtle molecular phenomena.

5.2 Introduction

The accumulation of genomic data has fueled the development of numerous computational tools to infer functional behavior from genomic sequences, thereby greatly expanding our understanding of how functional information is encoded in nucleic acid and protein sequences. Molecular biologists now have easy access to a plethora of sequence analysis tools that predict particular functional features from input sequences (Bailey et al., 2009; Giardine et al., 2005; Kingsford et al., 2007; Markham and Zuker, 2008; Thomas-Chollier et al., 2011). Common tasks comprise the identification of sequence motifs and functions from DNA/RNA or protein sequences (e.g., promoter or termination activity, recombination or splicing sites), as well as the computation of features that have been linked to particular phenotypes (e.g., codon usage, propensity to form transmembrane protein domains).

While such predictions are often used for biological discovery, the same tools can also be used to design new sequences that satisfy predefined properties of inter-

est. For example, codon optimization algorithms that generate sequences conforming to the codon usage of a host genome have been repeatedly shown to improve the expression of heterologous genes (Welch et al., 2011). However, most molecular behaviors result from the combined influence of several sequence determinants and, conversely, a single feature might impact several molecular phenomena. The multidimensional examination of DNA sequences is, therefore, necessary to capture the inherent complexity of biological processes, and further enable predictive design.

The exponentially decreasing cost of large-scale DNA synthesis (Carr and Church, 2009) opens the possibility to explore multiple genetic determinants of cellular phenotypes at high-throughput rates. It is currently within our capabilities to synthesize millions of short sequences on a single chip and further quantify the resulting phenotypes in living cells (Quan et al., 2011). Given these advances, it becomes possible to generate libraries that vary multiple sequence features in controlled ways so as to enable effective dissection of their individual activities and interactions. Such approaches generally follow a two-step process. In the first step, key sequence features potentially impacting a set of phenotypes of interest are identified from the analyses of available natural sequences. In the second step, a library is designed so that the identified features are systematically varied across the synthetic sequences to conform to statistical experimental designs that enable the precise estimation of the main effect of each feature and its interactions with others (Mutalik et al., 2013). Since it is often currently impossible to forward design sequences with desired features, it becomes necessary to generate randomized variants and use the stage-one analysis to select for sequences with the right combinations. Valuable tools taking multifactorial complexity into account and providing specialized multi-objective optimization solutions have been developed for specific applications (e.g., for protein synthesis optimization (Gaspar et al., 2013; Gaspar et al., 2012; Salis et al., 2009; Welch et al., 2011)). However, a general software for the constrained, stochastic optimization of sequence libraries that varies arbitrary sequence features in such a way that it is possible to isolate their effects and interactions does not currently exist.

Here, we present D-Tailor, an extendable framework that allows for the integration of multiple sequence analysis tools to mine and further design biological sequences in a single platform. D-Tailor provides a flexible and efficient sequence de-

sign algorithm to generate a single sequence optimized according to multiple features constraints or libraries of sequences satisfying particular design of experiments. Considering that the extent of molecular and functional interactions between sequence features is usually unknown, we demonstrate how our software can achieve systematic and balanced sampling of the feature space, thereby tailoring sequences to conform to full-factorial experimental designs that are ideal for the detailed characterization of complex genetic traits.

5.3 D-Tailor

D-Tailor essentially implements the two-step process described above (Figure 5.1).

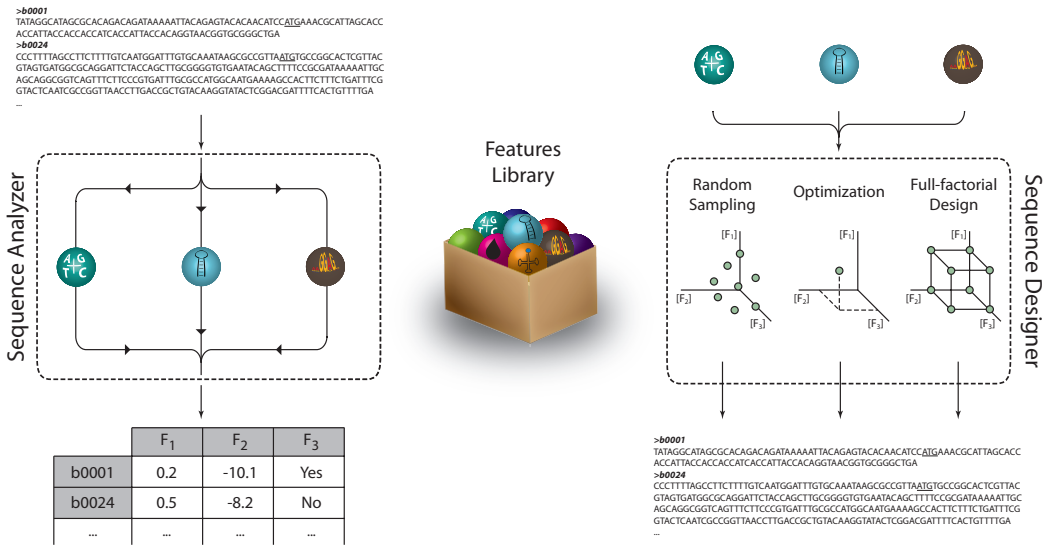


Figure 5.1 D-Tailor enables automated analysis and design of DNA sequences
D-Tailor provides a flexible and extendable architecture to explore different sequence features (box with multiple modules represented as balls). The left panel depicts an example of the retrieval process of three features from DNA sequences (nucleotide content evaluation, RNA structure prediction and sequence motif occurrence, respectively). The resulting table features profile is used to identify general trends and further define ideal ranges for the parameters values in the design mode. The right panel shows the design mode of D-Tailor, wherein multiple features are considered to design the final synthetic sequences based on multiple user-defined goals. Designed sequences can be generated randomly to simply probe the feature space (random sampling), or specifically targeted to match user-defined features scores (optimization and full-factorial).

The analysis mode performs feature evaluation from plain biological sequences. Here, the user specifies input sequences from which a selected set of features will be computed. The design mode of D-Tailor wraps the analysis routines in a parameter-

izable evolutionary algorithm that generates DNA sequences matching the specified features of interest. In a typical workflow, users can use the analysis mode to identify the features and values that are worth searching for in the design mode.

5.3.1 Brief implementation overview

D-Tailor provides an integrated framework for the seamless extraction of multiple features of interest from plain DNA sequences. This analysis pipeline was also integrated in a Monte-Carlo architecture that is used to evolve input sequences under user-defined constraints toward a set of target combinations of features scores, thereby enabling multi-objective sequence design. D-Tailor is based on an extendable architecture to allow the independent development of new sequence features evaluators that can be easily plugged-in to the software (Figure 5.1).

D-Tailor uses object-oriented design and its core entities are:

- *Feature* — abstract class encapsulating all relevant attributes and methods to describe a particular sequence feature or property;
- *Solution* — concrete class containing all information concerning a particular sequence.

A *Solution* can have one or more *Feature* objects. *Solution* is a concrete class that can readily be used to store a DNA/RNA sequence and perform subsequent extraction of the features of interest. In contrast, *Feature* is an abstract class that must be extended to implement a concrete feature. D-Tailor comes packaged with many ready-to-use concrete features that extend the abstract class *Feature*. Appendix I contains a detailed implementation of two of these features to exemplify how developers can also easily implement their own feature of interest.

Figure 5.2 depicts the unified model language (UML) class diagram capturing the dependencies between classes implemented in D-Tailor. The two main executable classes of D-Tailor are called *SequenceAnalyzer* and *SequenceDesigner*. These classes implement the basic functionality of the engine responsible for the analysis and design of sequences, respectively. These are abstract classes and need to be extended to implement user-defined analyses and designs (e.g., which features to compute or what region of the sequence can be mutated). The design mode requires the instantiation of an additional class that defines the design goal. This class must be

an extension of the abstract class *Design*. Several design methodologies are already implemented in D-Tailor (see Appendix I).

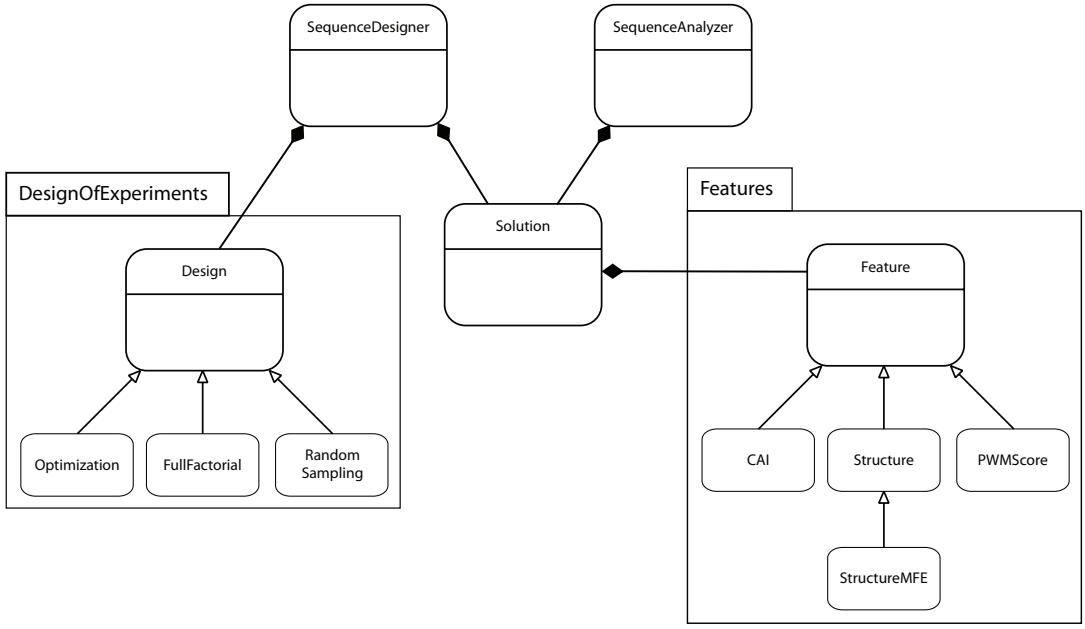


Figure 5.2 D-Tailor UML class diagram
This diagram depicts the main classes implemented in D-Tailor. The two main programs *SequenceAnalyzer* and *SequenceDesigner* contain one or more instances of the class *Solution*, which contains a list of one or more instances of the class *Feature*. The *SequenceDesigner* also requires an instantiation of the class *Design,* which provides basic information about the target(s) one is trying to design for. The diagram also shows some examples of classes extending the abstract classes *Design* and *Feature*.

To provide a convenient working environment and further enable parallel computation, results need to be permanently stored and easily accessed as sequences are generated and evaluated. D-Tailor implements a storage environment based on SQLite (http://www.sqlite.org/). However, other database solutions can be implemented to provide a storage environment compatible with the user preferences (e.g., SQLServer or MySQL) without impacting the basic functionalities of D-Tailor.

Appendix I contains a tutorial with detailed information about the implementation of D-Tailor in python, usage examples and development guidelines for creation of new feature modules extending current D-Tailor capabilities.

5.3.2 Sequence analyzer

Despite its apparent simplicity, the four-letter code provided by DNA chemistry can encode complex biological information. Fortunately, many functions can be readily inferred from the DNA sequence using available knowledge. For example, we can find a Shine-Dalgarno (SD) sequence in a prokaryotic 5' untranslated region (5'UTR) by identifying for the region immediately upstream of a start codon with the largest complementarity to the 3' end of 16S rRNA. The free energy of this complex can then be used as a rough estimate of the potential translation initiation rate.

D-Tailor provides a wrapper for multidimensional interrogation of DNA sequences. As such, it constitutes an improvement over current practices, which often require manual utilization of disperse software tools and can therefore be extremely laborious and error-prone. D-Tailor is designed with a modular architecture such that future developers can easily implement new modules for any particular feature of interest using custom code or scripts connecting to third party software.

As proof of concept, D-Tailor implements the automated extraction of several features that are known or presumed to impact gene expression throughout the central dogma of molecular biology. We chose these because there is a vast knowledge about how different DNA/RNA sequence features can dramatically affect transcriptional or translational regulation. D-Tailor comes out-of-the-box with 14 different modules including: score promoter regions or transcription factor binding sites based on sequence logos, estimate translation initiation rates based on SD sequences, predict propensity of a sequence to form RNA structures, calculate nucleotide composition, or evaluate the codon adaptation index (CAI) of a gene. These examples illustrate diverse implementation modalities and can be used to guide future software extensions.

In analysis mode, D-Tailor reads a set of sequences in either delimiter-separated or FASTA format files and computes a set of user-defined features for each sequence (see Appendix I). The output feature profile is a list that can be either written to file or directed to the standard output (Figure 5.1, left panel).

5.3.3 Sequence designer

The most innovative feature of D-Tailor is the capacity to design new sequences that target multiple features of interest (Figure 5.1, right panel). Current practices to design sequences with desired properties can be extremely complex and time-consuming, especially the features of interest overlap in the sequence space.

The design process in D-Tailor starts with the definition of a set of features, the specification of a design goal in that feature space and an initial sequence (seed) from which the designed sequences will be derived. Design goals can range anywhere between a single target combination of feature values to a comprehensive full-factorial design for a library of sequences. The specification of design targets requires discretization of the feature space into levels (defined by feature values intervals) to yield finite full-factorial designs. To ensure biological relevance, natural feature profiles extracted from available genomic sequences can be used to guide parameterization of these levels (see below). For each feature, users may define as many levels as necessary to attain the desired degree of resolution in the designed sequences. Nevertheless, given that the number of potential targets increases geometrically with the number of features/levels, their definition must consider downstream experimental capacities.

Finding a sequence that conforms to an arbitrary combination of features values (scores) is often computationally infeasible using brute force. Indeed, the sequence space to be searched is gigantic (4^N where N is the number of nucleotides in the sequence to be designed). In this context, it becomes necessary to use heuristics that can guide the designed sequence search towards the desired target. D-Tailor implements Monte-Carlo simulations to evolve a given seed sequence towards any design goal (i.e., one or more combinations of features scores).

More specifically, the algorithm loops through cycles of evolution until it finds all target combinations of features scores specified by the user (Figure 5.3). Each evolution cycle consists of three steps: i) a target feature combination is selected; ii) the repository of sequences previously generated (including the seed sequence) is searched to select a template sequence amongst the closest to target in the feature space (proximity check); iii) a defined number of mutational iterations starting with the selected template sequence is performed.

In turn, each of these mutational iterations also comprises 3 steps: i) the sequence being evolved is analyzed and a feature requiring optimization (i.e., not within the target level) is randomly selected; ii) the template sequence is then mutated according to user-specified mutational rules (see below); and iii) the features scores of the resulting sequence are analyzed and evaluated with respect to the current design target. If the new combination matches the target then the sequence is validated, the target is marked as completed and the evolution cycle is terminated. Otherwise, the algorithm chooses a template, either the current template or its mutated derivative, to be used for the next mutational iteration. At this point, we provide two predefined selective regimes: i) directional selection, wherein the sequence that is closer to the design target in the feature space is chosen; or ii) neutral selection: any of the two sequences is selected with equal probability.

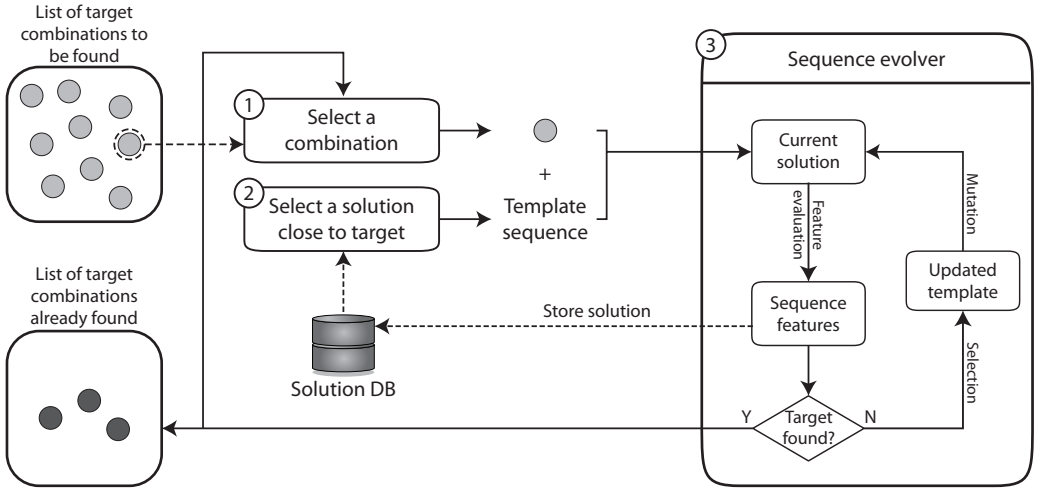


Figure 5.3 D-Tailor design algorithm
The main design algorithm is devised in three different steps: (1) selection of a target combination that we wish to design for; (2) select a template solution from the database (DB) that is close to the target combination in the feature space; (3) perform multiple mutations until a sequence with the desired combination is found or a maximum number of iterations is achieved.

At each of the mutational iterations, new sequences are generated through random mutation of the template sequence. To further improve the likelihood of generating changes that effectively lead to the desired target combination, D-Tailor implements targeted and, when available, oriented mutational operators. The former operators restrict the mutational space to specific regions of the sequence that are more likely to change the feature score, whereas the latter operators further define

specific mutation patterns to impact the feature score in the desired direction. For example, when the design goal calls for increasing the CAI of a gene, the targeted mutational operator specifies that the region to be mutated is comprised between the start and stop codons (that way preventing unnecessary and potentially deleterious mutations, for example, in the 5'UTR) and randomly replaces a codon irrespective of its CAI score. In addition, the oriented mutational operator chooses a codon and replaces it with one associated with a higher CAI score, thereby increasing the overall CAI of the gene. Any mutational operator targeting gene sequences can be further constrained to only generate synonymous mutations, thereby preserving the encoded protein sequence while modifying the underlying DNA features.

By manipulating the strength of selection and mutational patterns — two essential parameters of the evolutionary design process — users can indirectly control the extent of sequence diversification in the generated sequences and the overall performance of the target search (see below).

5.4 Results and discussion

As a representative example, we used D-Tailor to explore three different sequence determinants of translation efficiency in *E. coli*. Translation efficiency is not only determined by the overall elongation rate of a gene, which is roughly captured by its CAI (Sharp and Li, 1987), but also by the rate of translation initiation. In prokaryotes, translation initiation is known to be mainly determined by two factors: i) the presence of the SD motif, which increases the affinity between ribosomes and the transcript (Shine and Dalgarno, 1975); and ii) the strength of putative mRNA secondary structure in this same region, which hinders the former process (de Smit and van Duin, 1994). Importantly, the sequence defining the SD motif influences the formation of secondary structures. Likewise, secondary structure can be affected when modifying the codon usage at the beginning of the gene. This pattern of interlaced features makes their individual contributions to translation efficiency particularly difficult to untangle (Tuller et al., 2010) (Figure 5.4). As mentioned before, recent advances in DNA synthesis and cloning make it possible to dissect such complex relationships by generating synthetic sequences that best allow to investi-

gate the main effects and interactions between features of interest (e.g., full-factorial designs (Mutalik et al., 2013)).

With this perspective, we first used D-Tailor to analyze these three features across the entire *E. coli* genome (Figure 5.4). Such bulk analysis provides a solid basis to identify trends in the features of interest and identify the relevant parameter space to explore, as well as the thresholds to consider when discretizing continuous feature scores into levels (see Appendix I for details). Here, we defined five different levels for each feature based on their respective quintiles (Figure 5.4A, dashed lines). A full-factorial design based on such configuration yields a total of 125 (5^3) different target combinations of all feature levels across the three variables.

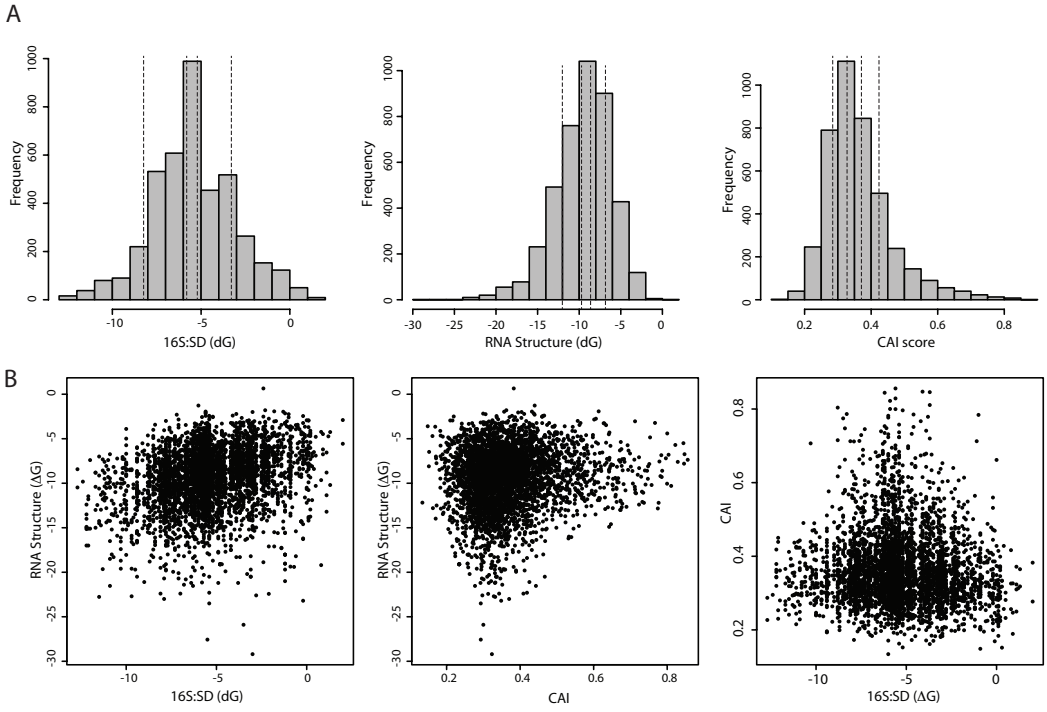


Figure 5.4 Using D-Tailor to analyze three different features across *E. coli*
(A) Distribution of the three different sequence features scores influencing translation efficiency in *E. coli*. The dashed lines indicate the quintile boundaries for the scores of each feature (hybridization energy between the 16S rRNA and SD sequence, minimum folding energy of RNA structure in the translation initiation region, and codon adaptation index of the gene sequences). (B) Scatter plots showing the cross-correlation between the three features of interest.

We next randomly selected 30 gene sequences along with their 5'UTR from *E. coli* to serve as seeds and compared four different design strategies within D-Tailor to generate a set of sequences conforming to a full-factorial design for each case

(Figure 5.5). We specified that designed sequences could be generated by unrestricted mutations in the entire 5'UTR region, composed of 49 nucleotides, but only synonymous mutations were allowed in the gene coding sequence.

We first used the most rudimentary design strategy available in D-Tailor, random sampling, to generate random sequences until the 125 different targets were found. Every attempt to complete this design goal using this purely random procedure was aborted after 3,000 generated sequences due to its obvious inefficiency (Figure 5.5, black line, 54.2 generated sequences per target found [gspt] on average).

The second design strategy included the canonical heuristic algorithm implemented by D-Tailor and used the simplest mutational method, wherein new sequences are consecutively generated by random mutation (Figure 5.5B, yellow line). This strategy significantly improved the efficiency of the search algorithm as compared to that of the random sampling method (24.8 vs 54.2 gspt on average, Mann-Whitney test p-value = 2.3×10^{-10}, Figure 5.5C). Nonetheless, the overall performance of the algorithm was still modest since many sequences had to be generated to find the required targets.

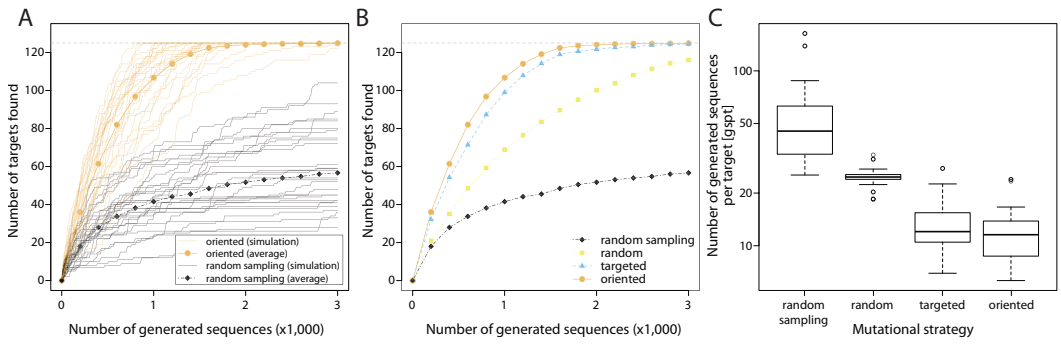


Figure 5.5 Mutational strategies performances
(A) Number of combinations found (out of 125, dashed line) as a function of the number of generated sequences (up to 3,000). Two different mutational strategies are depicted: oriented (orange) and random sampling (black). For each strategy, we performed 30 different simulation of a full-factorial design (faded lines) using different seed sequences. The solid lines represent the average number of target combinations found (across 30 simulations) as the number of generated sequences increases. (B) The average performance of the four different mutational strategies. (C) The number of generated sequences per target combination found using the different mutational strategies (n=30).

The third mutational strategy remarkably improved the search algorithm efficiency by employing spatially targeted mutations that more rapidly evolve a sequence to-

wards some desired features scores target (Figure 5.5B, light blue line). This strategy only required 13.3 gspt on average (Figure 5.5C).

Lastly, a fourth strategy using more 'rational' mutational operators that orient mutations toward the desired target provided slightly faster dynamics (Figure 5.5A-B, orange line). This incremental refinement to the mutational process resulted in a marginal improvement of the search algorithm performance (11.8 vs 13.3 gspt on average, Mann-Whitney test p-value = 0.129, Figure 5.5C).

In full-factorial designs, the number of potential combinations grows exponentially with the number of features and levels per feature considered. Therefore, we encourage the use and implementation of targeted and/or oriented mutational operator whenever possible, since these provide significant performance improvements.

For the two main mutational methods, the average dynamics across the different seeds (Figure 5.5B, orange and light blue lines) revealed that the rate at which new targets are discovered is relatively steady for the first ~80% of targets and slows down by the end of the search process, presumably because these targets have combinations of features scores harder to attain. We also observed that the dynamics of the desired targets discovery varies slightly depending on the input seed (Figure 5.5A, faded lines). This discrepancy between seeds demonstrates that different seeds can have distinct evolutionary landscapes and that some sequences may be more amenable than others to generate widely variable profiles of features scores with fewer mutational cycles (Cambray and Mazel, 2008; Wagner, 2008).

In the process of designing synthetic sequences with specific features scores, users may often want to limit the divergence of the designed sequences. To roughly control the spread of the generated sequences, users can tweak the strength of selection toward the desired targets. To better illustrate this point, we evolved the initial 30 seed sequences toward six different target combinations bearing different distances to the seed in the feature space (Figure 5.6A-B). Then, we examined the behavior of the algorithm in response to two contrasted selective regimes: neutral and directional selection.

As expected, we observed that a more relaxed selection process (neutral) is able to generate sequences matching the desired target that are more similar to the seed sequence (average hamming distance was 20.8, Figure 5.6C) as compared to those generated using the directional selection approach (average hamming distance was

30.2, Figure 5.6C) (Mann-Whitney test p-value = 0.0004). Nonetheless, this reduction in sequence diversity comes at the expense of longer sequence searches. In fact, for the 30 seed sequences tested, the neutral selection process required ~8-fold more gspt than the directional selection approach (Figure 5.6D). Therefore, users have to establish a compromise between the desired sequence divergence of the designed sequences and the computational/time feasibility of generating multiple sequences per target. Sometimes it may also be useful to use a hybrid approach wherein the algorithm is initially set with weak selection and hard constraints to avoid large sequence degeneration, and progressively configured with increased selection bias and/or relaxed mutational constraints (e.g., allow non-synonymous mutations in coding sequences) as the rate of target discovery slows down.

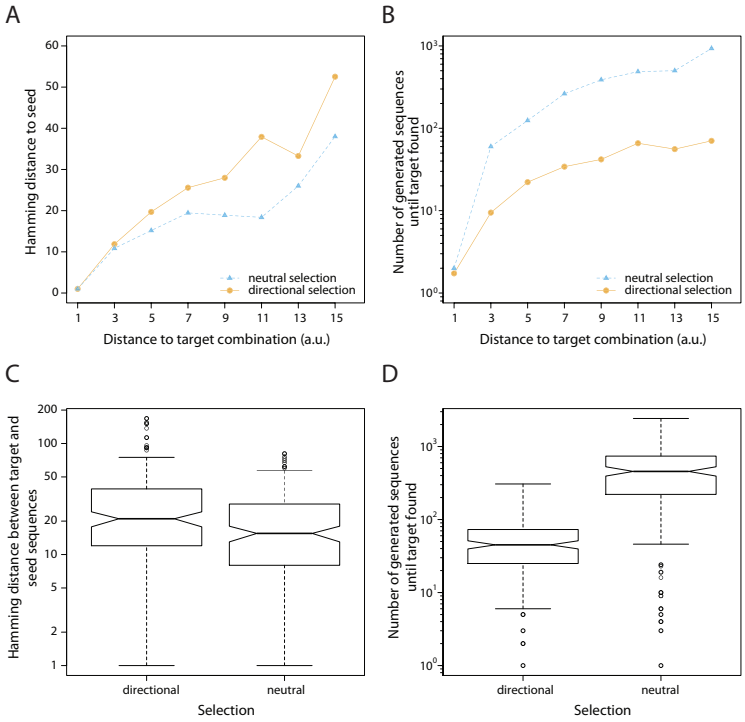


Figure 5.6 Selection options in D-Tailor
(A) The different lines show the average hamming distance between the seed and the sequence matching the target combination as a function of the distance (in the feature space) to the target combination using neutral (light blue) or directional (orange) selection. (B) The number of generated sequences until the desired target is found as a function of the distance (in the feature space) to the target combination using either neutral (light blue) or directional selection (orange). (C-D) The hamming distance (C) and number of generated sequences until target is found (D) for the 30 different simulations using either directional or neutral selection.

Considering ever increasing throughput in both synthesis and screening technologies, D-Tailor enables unprecedented large scale and hypothesis-driven design of artificial sequences that will aid dissect complex multifactorial phenotypes, and eventually derive quantitative mapping between sequence features and phenotypes. From a more theoretical perspective, D-Tailor provides a consistent framework to study the interplay between features and underlying primary sequences. A better understanding of such mapping will improve the design process and shed light onto the robustness and evolvability of functional biological sequences.

Chapter 6

Transcript Level and Sequence Determinants of Protein Abundance in *Escherichia coli*

This chapter is based on the article:

Joao C Guimaraes, Miguel Rocha and Adam P Arkin. Transcript level and sequence determinants of protein abundance in *Escherichia coli*. *In preparation*.

6.1 Abstract

Proteins mediate the majority of cellular processes that determine the physiological state of the cell and their abundance is a key measure of the corresponding activity. We performed a comprehensive analysis of more than 100 sequence features to derive a predictive model composed of a minimal non-redundant set of factors explaining 66% of the total variation of protein abundance in *Escherichia coli*. The model suggests that protein abundances are primarily determined by the transcript level (53%) and effectors of translation elongation (12%), whereas only a small fraction of the variation is explained by translational initiation (1%). Our analyses uncover a new sequence determinant not previously described affecting translation initiation and suggest that elongation rate is affected by both codon biases and amino acid composition. We also show that transcription has the largest effect on the dynamic range of protein abundances, whereas translation efficiency has a significant effect on cell-to-cell variability of protein levels.

6.2 Introduction

Protein production can be energetically costly for the cell and constitutive expression levels and their regulation is thought to have evolved to meet the tradeoff between cost and utility (Gingold and Pilpel, 2011; Vogel and Marcotte, 2012). While protein abundance is a better proxy for protein activity — usually the quantity of interest if one wishes to understand the current capability of a cell — it has become common practice instead to use mRNA transcript abundances since these are far easier to measure at genome scale. Nonetheless, many studies in bacteria demonstrate that transcript abundances are only moderately correlated with protein abundances (coefficient of determination $R^2 \sim 0.17\text{-}0.47$) (Lu et al., 2007; Maier et al., 2009; Maier et al., 2011; Masuda et al., 2009; Nie et al., 2006). Hence, greater than 50% of the variability of protein abundance across the genome must be explained by post-transcriptional processes that effect translation efficiency and protein degradation though we will not discuss the latter here (Greenbaum et al., 2003). Additionally, some of these parameters may as well be intertwined in com-

plex ways — for instance, high overall translation efficiency can lead to a higher density of ribosomes protecting the transcription from degradation and thereby affect absolute transcript abundance (Deana and Belasco, 2005). For these reasons, it becomes crucial to dissect the individual contribution of the multiple mechanisms to the steady-state level of proteins.

For endogenous genes of prokaryotic organisms, it is generally believed that translation initiation is the rate-limiting step of protein synthesis (Andersson and Kurland, 1990; Bulmer, 1991). This assertion is supported not only by the biophysics of translation (Bulmer, 1991), but also by the observation that ribosomes in a polysome are fairly spaced and by the lack of correlation between experimental data on the number of ribosomes bound to a particular messenger RNA and its translation efficiency (Belle et al., 2006; Beyer et al., 2004; Brockmann et al., 2007; Plotkin and Kudla, 2011). Intuitively, it also looks more efficient to modulate the expression level of a gene by tuning the efficiency of a promoter and/or a ribosome-binding site rather than altering multiple codons of a gene to tweak its elongation rate. Nonetheless, the non-random utilization of the different synonymous codons (i.e., those encoding the same amino acid) is pervasive in nature (codon bias). The natural selection theory (Bulmer, 1991) posits that such biases result from the adaptation to tRNA pools and are more noticeable in highly expressed genes because these are subject to a greater pressure for translation accuracy (Akashi, 1994) and efficiency (Ikemura, 1985). Though there is some evidence that codons are translated faster by more abundant cognate tRNAs (Varenne et al., 1984), large-scale studies measuring endogenous mRNA and protein levels have failed to show a correlation between translation efficiency and codon bias (Lu et al., 2007; Vogel et al., 2010). Additionally, a recent study using 154 synonymous genes encoding the same fluorescent protein showed that the formation of RNA structure inhibiting initiation, rather than codon bias, was the main determinant of protein synthesis rate (Kudla et al., 2009). Altogether, these results are consistent with the hypothesis that initiation is a rate-limiting step, while ribosome density and codon usage only have a marginal effect on translation efficiency (Bulmer, 1991).

Here, we investigate the combined influence of mRNA abundance and more than 100 transcript sequence features, believed to control translation initiation and elongation efficiency, on genome-wide protein levels of *Escherichia coli*. We devel-

oped an integrative model to find a minimal set of sequence features highly predictive of protein abundances on unseen data (via cross-validation). The model, comprised by 16 predictors, explains 66% of variation of protein abundance genome-wide. We show that translation efficiency is affected by general codon bias, as well as the usage of a few specific codons and amino acids. The latter preferences are in good agreement with the abundance of the respective tRNA pools. We also found that, in contrast to the arguments above, determinants of translation initiation only explain a small fraction of the total variation of protein abundance (~1%) and we report a new feature of the translation initiation complex that may be responsible for the efficient dissociation of this complex and consequent start of elongation step. Finally, we demonstrate that our estimates of translation efficiency show a positive correlation with the amount of cell-to-cell variability in expression levels as expected from translational burst behavior.

6.3 Materials and methods

6.3.1 A predictive model of protein abundance and feature selection

To select an explanatory model of protein abundance (PA) built from tens of possible predictors we used partial least squares (PLS) regression. PLS is a method for relating two data matrices (X, a matrix with multiple predictors and Y, a matrix with response variables), by a multiple linear regression model. PLS finds a dimensionally-reduced projection of X that captures most of its variance and has a maximum covariance with Y. This is the method of choice for handling multicollinearity among X values and hence provides a more robust estimation of regression coefficients than simple multiple linear regression method. The following equation shows the linear relationship between response variable and predictors where the factor interactions were excluded due to the difficulty in biological interpretation of these terms and because, when included, they did not significantly improve the model performance:

$$\log(PA) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n + \varepsilon \qquad \text{Eq. 6.1}$$

where x_i are the multiple predictors (mRNA concentration and sequence features), β_i are the regression coefficients for each of the explanatory variables (i = 1, ..., n), β_0 is the regression constant, and ε is the error term. Numeric predictors and response variable were converted to a normalized standard score (z-score) and regression models were obtained using the package *pls* (Mevik and Wehrens, 2007) for the R software suite (Team, 2011).

We then employed stepwise regression with backward selection to downselect the initial 108 predictor variables (see Appendix II) to a final set of 16 showing the highest explanatory power (Figure 6.4). More specifically, we generated composite models with less complexity by iteratively removing the variables that did not significantly reduce the predictability of the model, which was evaluated using a 10-fold cross-validation (CV) procedure. The variables to be removed were selected based on the significance of the jackknife variance estimates for the regression coefficients calculated from the 10-fold CV.

The performance of the best PLS model was also compared with that of simple multiple linear regression, as well as that of non-linear models, namely neural networks (NN), support vector machines (SVM) and random forest (RF). These models were fitted using the algorithms implemented by the package *rminer* (Cortez, 2010) for the R software suite.

6.3.2 Data sources

We used the transcript and protein abundances for 824 genes obtained by RNA-seq and protein fluorescence-fusion measurements, respectively, collected from *E. coli* strain W3110 grown on M9 media and acquired during exponential phase (Taniguchi et al., 2010). This is the first genome-wide quantitative dataset for both mRNA and protein abundances with single cell resolution in *E. coli*. This dataset is particularly interesting because it provides cell-to-cell variability (expression noise) for each of the measured proteins. We retrieved the corresponding genome from GenBank (http://www.ncbi.nlm.nih.gov/nuccore/NC_007779) and used it to compute sequence-related features impacting gene expression. Aberrant genes containing frame-shifts or non-sense start codon were removed from final analysis.

We also evaluated the linear association between the mRNA and protein abundance to find genes with extreme deviation from the assumed relationship (Figure 6.1). We found 13 genes with extreme post-transcriptional regulation (residual variance greater than 3σ) and that for some of them had already been described complex regulation mechanisms that fall outside the scope of this study (e.g., small RNA inhibition). Therefore, we decided to remove these genes from the final analysis.

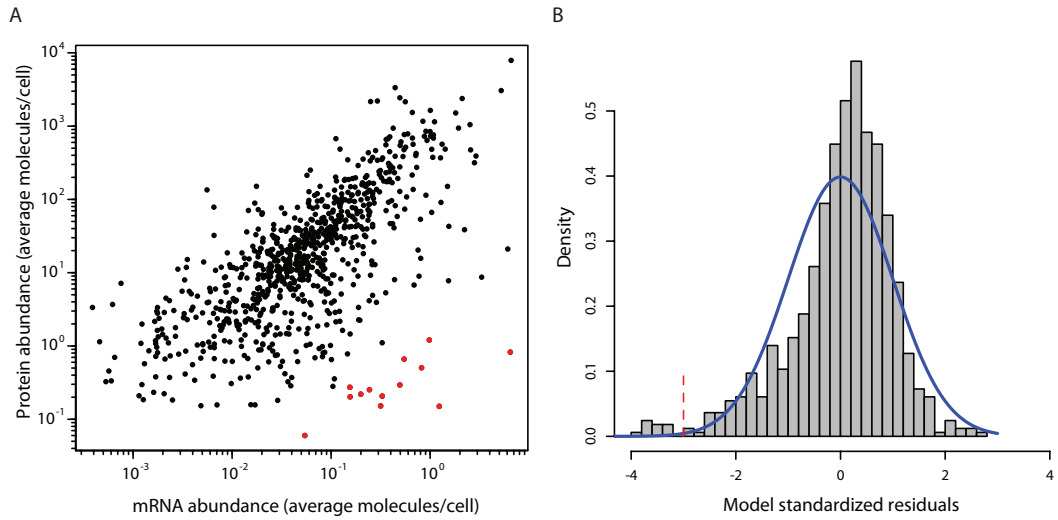


Figure 6.1 Association between mRNA and protein abundance
(A) The plot shows a very strong correlation (Pearson correlation coefficient $r = 0.7252$) between the experimentally measured mRNA and protein abundances, where each point is a gene. Despite the evident association between mRNA and protein abundance, there are a few genes that diverge significantly from the fitted linear regression (highlighted in red). **(B)** Histogram of the standardized residuals for the linear regression between protein and mRNA abundances. The red dashed line indicates the 3σ deviation from the expected mean of the normal distribution of residuals centered in 0. There are 13 genes with residuals variance greater than 3σ, which are highlighted in red in panel (A). All of them show very high levels of mRNA when compared to the corresponding low protein abundance. Among these genes, we found some that are post-transcriptionally regulated by forming long-range mRNA secondary structures (*gnd*) that inhibit translation initiation (Chang et al., 1995), as well as two other genes (*dppA* (Urbanowski et al., 2000) and *ompC* (Chen et al., 2004)) that are *trans*-regulated by small RNAs. These observations point out that some of these genes may have very complicated regulatory mechanisms that our model does not intend to cover and hence were removed from the final analyzed dataset.

6.3.3 Sequence features

A total of 107 sequence features were computed from two different regions of the messenger RNA: translation initiation region (TIR) defined as the region between -25 and +30 with respect to the start codon, and the coding sequence (CDS)

defined as the region between the start and stop codon inclusive (see Appendix II). Sequence features within these two regions have been shown to influence translation initiation and elongation rates, respectively. Features considered in the TIR influencing translation initiation rate include the multiple characteristics of the hybridization complex between the 3' end of 16S rRNA and the Shine-Dalgarno (SD) sequence, identity of the start codon, distance between the SD sequence and the start codon, and formation of RNA structure (Chen et al., 1994; de Smit and van Duin, 1994; Hall et al., 1982; Kudla et al., 2009; Lee et al., 1996; Salis et al., 2009; Shine and Dalgarno, 1975; Studer and Joseph, 2006; Vellanoweth and Rabinowitz, 1992) (Figure 6.2 and Appendix II). In the CDS region, we selected features that are likely to impact translation elongation rate: start/stop codon identity, codon usage, amino acid usage, AT/A content, codon adaptation index (CAI) and protein length (Akashi, 2003; Allert et al., 2010; Eyre-Walker, 1996; Sharp and Li, 1987; Welch et al., 2009a) (Appendix II).

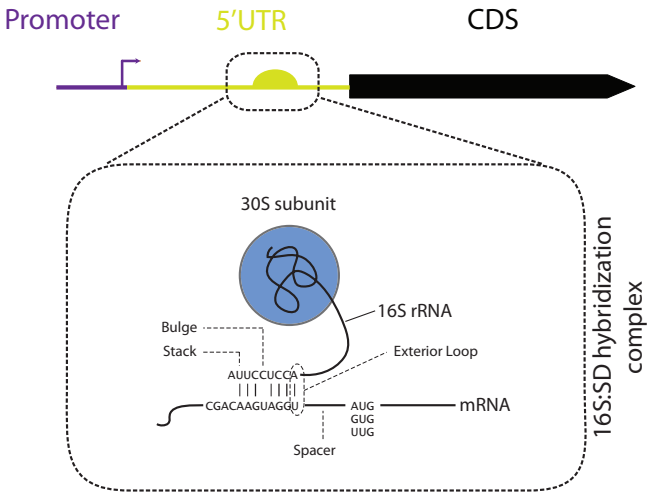


Figure 6.2 Sequence features of the 16S:SD hybridization complex
The figure depicts a detailed schematic of the sequence features considered for the 16S:SD hybridization complex.

Simulations of single and hybridized structures of RNA were performed using the UNAfold software (Markham and Zuker, 2008) and in-house Perl scripts were developed to extract relevant features from the predicted RNA structures. SD sequence motif for each gene was scored using Patser software (Hertz and Stormo, 1999) and the respective SD position frequency matrix from *E. coli* (Shultzaberger

et al., 2001). Details on the sequence features considered in this study can be found in Appendix II.

6.4 Results

6.4.1 Individual predictor performance

Cellular protein abundances result from the combined effect of multiple mechanisms that tune production and degradation. For example, the steady-state mRNA concentration of a gene is the combined outcome of transcript production and degradation, and, as expected, we identified a strong positive correlation between mRNA and protein levels (Pearson correlation coefficient $r = 0.7262$, P-value ≤ 0.001, Table 9). Additionally, to identify determinants of translational regulation we explored the impact of more than 100 sequence features on PA. These features were computed from two different regions of the messenger: translation initiation region (TIR) and the coding sequence (CDS) (Materials and methods). Sequence features within these two regions are believed to influence translation initiation and elongation rates, respectively.

Table 9 Factors individual correlation
List of the top five predictors with most significant partial Pearson correlation coefficients with protein abundance given the mRNA concentration for each category of features considered. F-test p-values were adjusted using (FDR) method (Benjamini and Hochberg, 1995) to correct for multiple testing: * P-value ≤ 0.05, ** P-value ≤ 0.01, *** P-value ≤ 0.001.

	Correlation with PA	Partial Correlation with PA given mRNA levels
mRNA level	0.7262***	0
TIR		
16S:SD (Exterior Loop ΔG)	0.1075**	0.1239***
RBS calculator score	0.1462***	0.1101**
Accessibility	0.0606	0.0912**
Single stranded bases	0.0630	0.0884*
Folding energy (ΔG)	0.0635	0.0729*
CDS		
CAI	0.5828***	0.3019***
ATC	0.3974***	0.2608***
GAA	0.3215***	0.2472***
Ile	0.1933***	0.2316***
Glu	0.2940***	0.2208***

We observed that many sequence features are moderately correlated with PA and slightly less partially correlated with PA given mRNA levels (Figure 6.3 and Appendix II). Bulmer and others have suggested that initiation is the rate-limiting step of translation (Andersson and Kurland, 1990; Bulmer, 1991). However, our results showed that sequence features related to initiation are generally less correlated with translation efficiency (i.e., PA given mRNA levels) than elongation determinants (Figure 6.3 and Table 9). A number of seminal (de Smit and van Duin, 1990; Hall et al., 1982) and recent studies (Kudla et al., 2009; Salis et al., 2009) have focused on the propensity of RNA structures to control translation initiation. We found significant correlations between many features related to RNA structure within the initiation region and translation efficiency (Table 9). As expected, our results suggest that weaker RNA structure in this region contributes to an increased production of protein. We also observed that a recently developed calculator of translation initiation rates (Salis et al., 2009), which consolidates several determinants of initiation such as RNA structure and SD sequence strength, is significantly correlated with PA given mRNA levels ($r = 0.1101$, P-value = 0.0017). Surprisingly, we found that the binding free energy — lower energy corresponds to tighter binding — of the external loop of 16S:SD hybridization complex is the initiation-related predictor with the highest positive correlation with translation efficiency ($r = 0.1239$, P-value ≤ 0.001), which suggests that weak binding at this particular region can be favorable for translation initiation (Figure 6.2).

The influence of codon bias on translation efficiency is currently topic of active debate. Several studies advocate that usage of codons adapted to tRNA increase protein yield (Cannarozzi et al., 2010; Gustafsson et al., 2004; Tuller et al., 2007; Tuller et al., 2010), while many others failed to find correlations between codon bias and translation efficiency (Ingolia et al., 2009; Kudla et al., 2009; Lu et al., 2009; Vogel et al., 2010; Welch et al., 2009a). Our results show that a genome-wide codon preference metric, CAI, is very significantly correlated with PA after controlling for mRNA abundance ($r = 0.3019$, P-value ≤ 0.001). Furthermore, we observed very significant correlations for the usage of specific codons (e.g., ATC: $r = 0.2608$, P-value ≤ 0.001 or GAA: $r = 0.2472$, P-value ≤ 0.001). Lastly, we also found a significant correlation between the abundance of a protein and the relative representation of some of its amino acids (e.g., Ile: $r = 0.2316$, P-value ≤ 0.001 or Glu: $r =$

0.2208, P-value ≤ 0.001). The importance of protein's amino acid composition has been observed for other prokaryotic (Nie et al., 2006) as well as eukaryotic organisms (Akashi, 2003; Greenbaum et al., 2002; Tuller et al., 2007; Vogel et al., 2010).

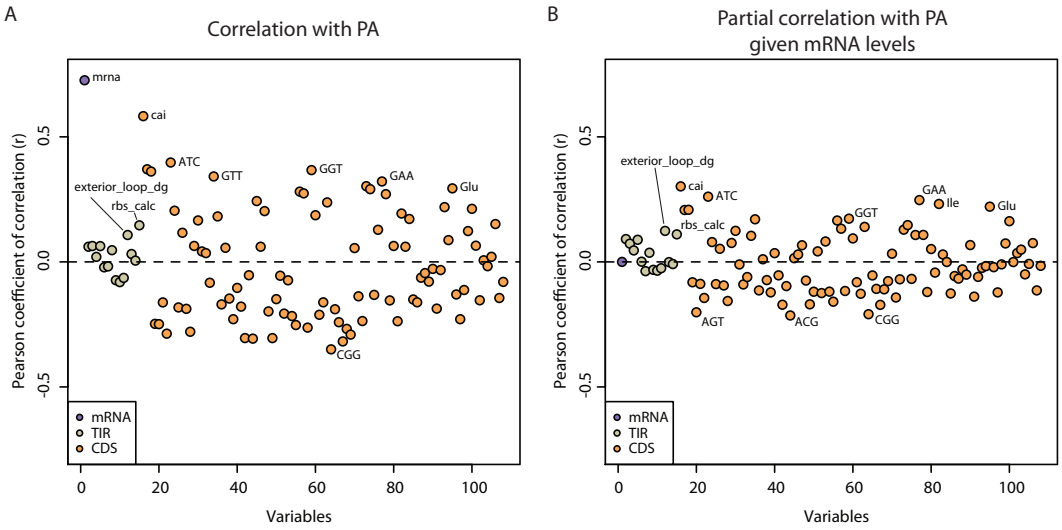


Figure 6.3 Correlation between features and protein abundance
The figure depicts the correlation between each of the 108 features considered and protein abundance (A), or protein abundance given the mRNA levels (B). This analysis demonstrates that most of the factors are moderately correlated with absolute protein abundance and slightly less associated with protein levels given mRNA abundances.

6.4.2 A composite model to predict protein abundance

We next sought to explore the combined effect of transcription- and translation-related features to predict the steady-state protein concentration across the whole genome. In contrast to previous studies that have employed similar approaches to predict protein abundance in a different bacterium (Nie et al., 2006), our method yields an integrated model based on a minimal number of explanatory factors that is validated using unseen data (Materials and methods). For that, we derived a PLS regression model where the mRNA abundance and sequence features are used as the explanatory variables and PA as the response variable. Then, given the large number of factors considered, we employed a stepwise regression method with backward selection to identify the predictors with higher explanatory capacity (Figure 6.4A and Materials and methods). We found that a PLS model considering only 16 predictors and 3 components showed the highest accuracy ($r = 0.81$, $R^2 = 0.66$ and cross-validated (CV) $R^2 = 0.65$, Figure 6.4B and Figure 6.5).

We also explored the possibility of using more complex models considering non-linear and interaction effects, but none of them showed better accuracy than the PLS model (Figure 6.6). Therefore, we selected the simpler composite model to study in greater detail the expression determinants of *E. coli*. Our model integrating mRNA levels and sequence features influencing translation efficiency explained 66% of the variability of protein abundances across the genome (Figure 6.5A). As expected, transcript abundance is the main determinant of protein concentration (53%), as it encompasses the result of several mechanisms of transcript production and stabilization. CDS sequence features likely controlling translation elongation stand out as the second most important explanatory class (12%), followed by a small yet significant contribution of translation initiation determinants (1%) (Figure 6.5B).

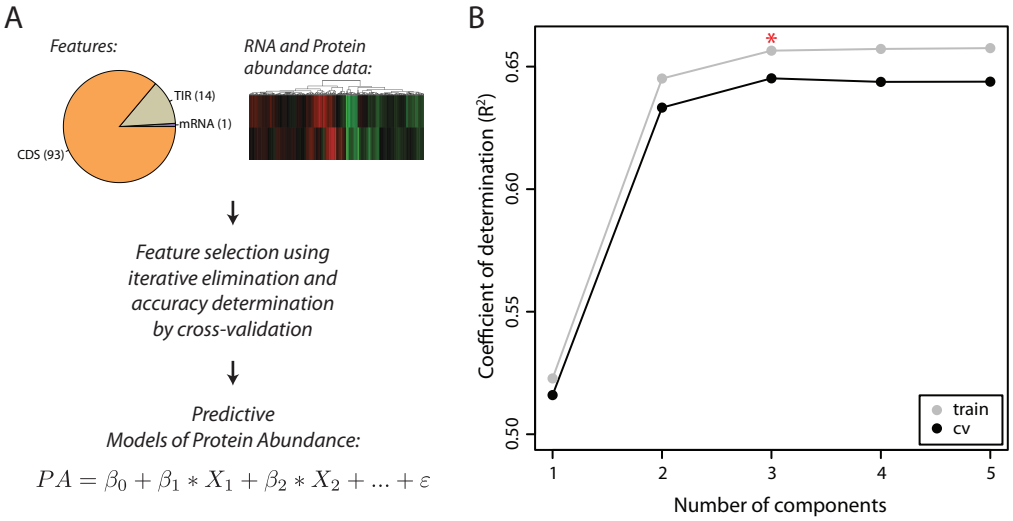


Figure 6.4 Determinants of protein abundance and composite model search
(A) More than 100 sequence features were integrated with mRNA levels to predict experimentally measured protein abundances. To select the best composite model, we employed a stepwise regression with backward selection to find the variables with the highest explanatory power (Materials and methods). (B) The plot depicts the PLS model performance — both using the entire training set (gray) and the 10-fold CV procedure (black) — as the number of components increases. A model composed of only 3 components (red asterisk) is enough to capture the maximum covariance between the 16 predictors and the response variable (protein abundance).

The regression coefficients of the linear model estimate the weight of each predictor on the steady-state protein concentration (Figure 6.5C). The contribution of mRNA level stands out as the dominant effect immediately followed by the CAI score. The only feature selected influencing translation initiation is the free energy of the exte-

rior loop of the 16S:SD hybridization complex (exterior_loop_dg). Its positive regression coefficient indicates that weak binding in this region is beneficial for translation efficiency. Surprisingly, in addition to CAI, which measures the overall codon adaptation of the gene, we found a set of codons that further influence the observed protein abundance, presumably by controlling gene's elongation rate. Specifically, codons GGT (Gly), ATC and ATT (Ile) have a positive effect, whereas TCA (Ser), ACC (Thr), TGT (Cys) and CGG (Arg) seem to be detrimental. These weights are in agreement with both the measured abundance of the corresponding tRNAs (Dong et al., 1996) (Table 4) and codon usage preferences in *E. coli* (Ikemura, 1985) (Table 3). Lastly, the prevalence of certain amino acids in protein's coding sequence can also enhance (Glu, Leu, His, Ile and Phe) or reduce (Met) protein abundance. Such contributions for Glu, Leu, Ile and Met were expected from early observations that amino acid usage correlates well the concentration of the respective tRNAs (Yamao et al., 1991). Perhaps more puzzling are the positive contributions resulting from the usage of amino acids His and Phe since these are infrequent in endogenous genes and present a high biosynthesis cost to the cell (Akashi and Gojobori, 2002).

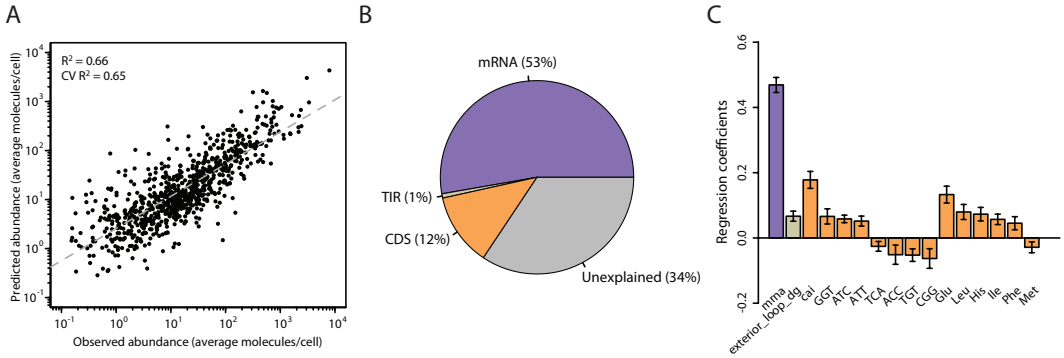


Figure 6.5 Determinants of protein abundance in *E. Coli*
(A) Predicted versus experimentally measured protein concentration using a composite model with 16 factors (R^2 = 0.66 and CV R^2 = 0.65). (B) Aggregated explanatory capacity of the factors in each group to predict variation of protein abundance. (C) Regression coefficients for all the factors in the model. Error bars represent the standard deviation of the regression coefficients based on jackknife variance estimates from 10-fold CV procedure.

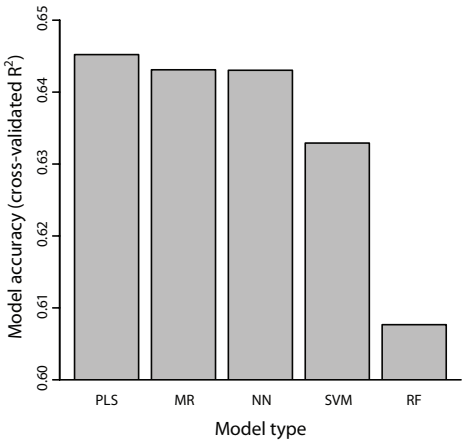


Figure 6.6 Partial least squares (PLS) regression model shows better accuracy than more complex models

We compared the accuracy of the PLS model with that of the regular multiple regression (MR), as well as more complex models such as: neural networks (NN), support vector machines (SVM), and random forest (RF). By using the latter models, we expect to encapsulate more complex behavior of our predictor variables, such as non-linearity and interaction between factors. These models were fitted using the data-mining wrapper *rminer* developed for R (Cortez, 2010). NN and SVM parameters were tuned to yield the best performing model, namely the number of neurons in the hidden layer and the Gaussian kernel parameter, respectively. The RF model was run using default parameters. We show that the PLS model employed in this study has higher accuracy than all the other models. This means that the assumed linear relationship between the predictors and the response variable protein abundance is acceptable and that considering non-linear behavior and interaction between these factors does not yield better results.

6.4.3 Expression profile of *E. coli* genes

The developed model integrates a set of predictors that can explain how protein concentration can be tuned at three different levels: transcription, translation initiation and elongation. Hence, we used the model to analyze the quantitative contribution of the different groups of predictors to produce distinct patterns of protein expression. For that, we split genes expression levels in three groups: low, medium and highly abundant genes using the lowest and upper quartiles. We then calculated the contribution (i.e., the weighted sum of all explanatory variables belonging to a class) of the two main classes of predictors (mRNA and CDS) to the steady-state protein concentration (Figure 6.7A).

We observed that mRNA concentration has a less/greater median contribution than the CDS features to the low/high abundant proteins. Further, the dynamic range of expression achieved by altering mRNA levels is slightly larger than by al-

tering CDS features. These two main determinants show a concerted effect to achieve the desired protein concentration (Figure 6.7B-D). For example, the expression of low abundance genes seems to be preferentially attained by expressing mRNA to low/medium levels and by CDS features that tend to correlate with lower expression, whereas highly abundant genes require both high expression and features encoding efficient translation elongation. The correlation pattern observed between transcription and translation efficiencies is expected given the fact that genes transcribed in greater abundance will also need to be rapidly translated to avoid depletion of free ribosomes in the cell.

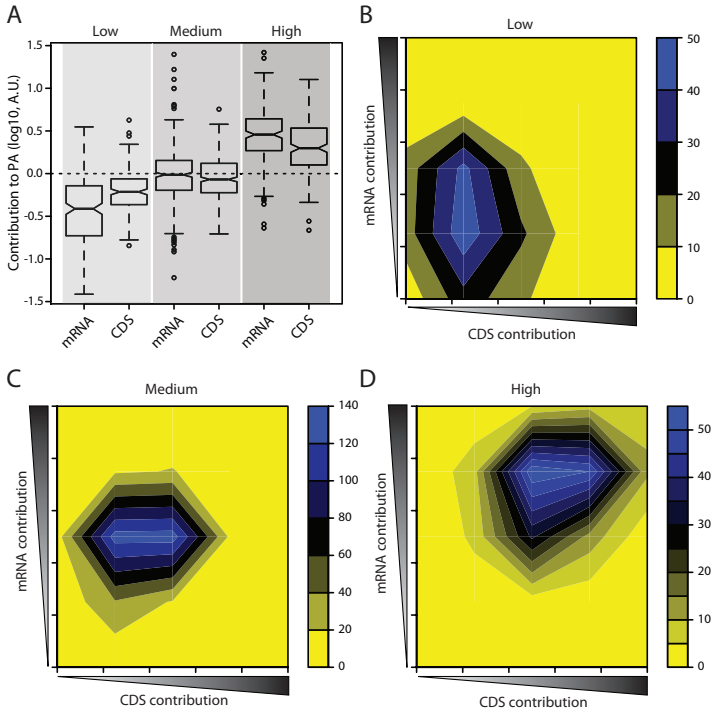


Figure 6.7 Transcription and translation efficiency act in a concerted fashion
(A) Individual contribution of mRNA and CDS features for low, medium, and highly expressed genes. We also observed a concerted contribution of mRNA levels and CDS features to the steady-state protein abundance. (B) Most low abundant genes tend to be expressed using medium to low levels of mRNA and a low contribution of the CDS features. (C) Genes expressed at medium abundance show a balanced cooperation between mRNA and CDS contribution, where both factors appear most of the time at average levels. (D) Highly abundant genes demand for high levels of mRNA and a medium-high contribution of CDS features.

The dataset (Taniguchi et al., 2010) used in our study provides absolute protein abundance with single cell resolution (Materials and methods) and therefore estimates of cell-to-cell variability of expression levels (noise). Gene expression is a sto-

chastic process wherein proteins are produced in sharp bursts and, as a consequence, enhanced translation efficiency increases cell-to-cell variation of protein abundances (Ozbudak et al., 2002; Thattai and van Oudenaarden, 2001). We sought to correlate the noise strength (defined as the Fano Factor, σ^2/μ) with our estimates of translation efficiency (Figure 6.8). This metric can be used to quantify the effect of translational burst described above, as this mechanism predicts that noise strength increases linearly with the average protein abundance when translation efficiency is increased. Practically, this means that if two genes are expressed at a similar average protein level, the one with highest translation efficiency will have larger noise strength and, hence, increased variability (i.e., a broader protein distribution). Indeed, for each group of expression level we found that for the set of genes showing the lowest and highest translation efficiency (based on quartiles) that the more efficient the translation the larger the expression noise. This phenotype change is more evident for medium and highly expressed genes (Figure 6.8). In contrast, less abundant genes do not show a significant alteration of cell-to-cell variability (Figure 6.8) most likely because translation efficiency of these genes remains fairly low (Figure 6.7B).

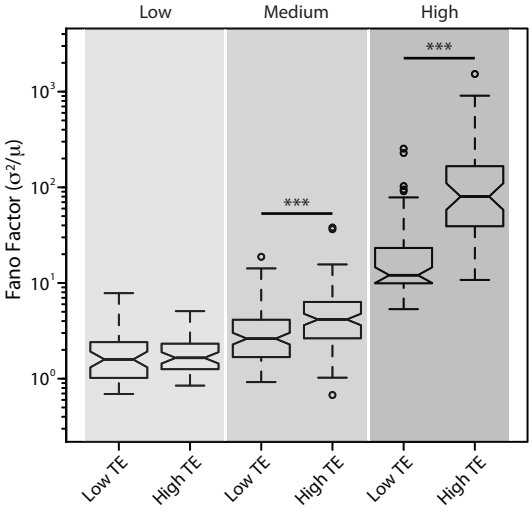


Figure 6.8 Translation efficiency affects protein cell-to-cell variability
The boxplot shows the cell-to-cell variability (defined by the Fano Factor) for protein being expressed at three different levels (low, medium and high) using disparate translation efficiencies (TE). Whereas for low expressed protein the cell-to-cell variability remains similar, for medium and highly expressed protein, enhanced translation efficiency significantly increases the expression noise. Mann-Whitney test significance: * P-value ≤ 0.05, ** P-value ≤ 0.01, *** P-value ≤ 0.001.

6.5 Discussion

Large-scale transcriptome and proteome measurements provide an invaluable source of information to interrogate the multiple determinants of steady-state protein levels. It is widely accepted that transcript levels are the main determinants of expression levels, however there is still a significant variation in protein concentration resulting from post-transcriptional regulation. Our results are in good agreement with this and indicate that 53% of total variation of protein levels can be explained by differential transcript abundances. We also estimated that 13% of the remaining variation is due to factors influencing translation efficiency.

It is generally believed that initiation is the rate-limiting step of translation and, as such, codon bias should only have a minor effect on translation efficiency (Andersson and Kurland, 1990; Bulmer, 1991). In equilibrium, the protein synthesis rate will be equal to the number of successful elongation termination events, which will be, at most, equal to the number of ribosomes that initiated translation. Therefore, a change in the rate of elongation can only lead to enhanced efficiency if it increases the rate of initiation (Bulmer, 1991). There are two mechanisms by which this may happen: 1) an overall increase in the speed of translation will make ribosomes flow faster through the transcript and, hence, become more rapidly available to the pool of free (Andersson and Kurland, 1990; Bulmer, 1991); and 2) codons translated faster will reduce the queuing of ribosomes and lead to more efficient initiation of translation (Bulmer, 1988; Liljenstrom and von Heijne, 1987; Tuller et al., 2010). Our results clearly indicate a significant impact of codon bias and amino acid usage on translation efficiency (Figure 6.5B-C), presumably by influencing the elongation rate. We are confident that these effects are directly associated to the adaptation to tRNA pools as confirmed by the regression coefficients estimated for the linear model. For example, the three codons with positive coefficients are recognized by two highly abundant tRNAs, whereas the four negative codons are recognized by four different tRNAs present in much smaller amount (~6% and ~1.5% of total amount of tRNA, respectively (Dong et al., 1996), Table 4). Regression coefficients are also in accordance with codon usage preferences of iso-accepting tRNAs (Ikemura, 1985) as the slight preference for ATC over ATT can attest (Figure 6.5C).

Translation initiation rate is influenced by many factors including: the affinity between the 16S rRNA and the SD sequence (Lee et al., 1996), the initiation codon (Ringquist et al., 1992) as well as the RNA structure formed in the initiation region (de Smit and van Duin, 1990; Studer and Joseph, 2006). Genetic alteration perturbing these elements can vary protein synthesis rates up to a 1,000-fold (Barrick et al., 1994; de Smit and van Duin, 1990; Lee et al., 1996; Salis et al., 2009).

Though we find that some of these determinants are significantly correlated with translation efficiency (Table 9), we are surprised to see that our integrative model only selected one initiation-related predictor able to explain ~1% of total variation of protein abundance. This modest predictive power may be justified by the multitude of translation mechanisms tolerated in *E. coli* as opposed to more conservative organisms such as *B. subtilis*, which only recognizes canonical initiation regions (Boni, 2006). Such versatility may hinder the identification of initiation-related determinants by simple linear models, such as the one used in this study, and may also justify the weak SD motif signal observed in *E. coli* endogenous genes (Shultzaberger et al., 2001). The translation initiation-related predictor selected by our model (exterior_loop_dg, Figure 6.2) suggests that the free energy of the external loop of the 16S:SD hybridization complex is positively correlated with efficient translation. Though such mechanism has never been identified and, therefore, future experimental confirmation is necessary, we speculate that weak binding in this region may have a beneficial effect on expression by facilitating the subsequent disruption of the ribosome from SD sequence to start the elongation stage. Our hypothesis is further indirectly partially supported by the fact that extremely long complementarity between 16S rRNA and SD sequence does not produce highest translation rates (Vimberg et al., 2007).

Lastly, we find a general correlation between our estimates of transcription and translation efficiency, which demonstrate the concerted operation of the two mechanisms (Figure 6.7). Genes that are highly transcribed create demand for more ribosomes and, hence, must be efficiently translated to avoid depletion of free ribosomes in the cell and maximize growth. We also evaluate the validity of our translation efficiency estimate by analyzing the cell-to-cell variability of expression levels. Stochastic gene expression posits that protein production results from random and sharp bursts and, as a consequence, increased translation efficiency will lead to

higher expression noise (Ozbudak et al., 2002; Thattai and van Oudenaarden, 2001). We show that our estimates of translation efficiency are in agreement with the hypothesized phenotypes (Figure 6.8).

There is still ~34% of variation of protein levels that is not explained by our model (Figure 6.5B) and may result from measurement variability (~15% as estimated from replicate to replicate variability (Taniguchi et al., 2010)), as well as other parameters not directly related to the general properties of translation studied here, but to gene-specific regulation (e.g., *trans*-regulation by small RNAs, Figure 6.1). Additionally, protein decay rates have also been shown to impact steady-state protein concentrations (Belle et al., 2006; Maier et al., 2011; Schwanhausser et al., 2011; Vogel et al., 2010).

Our analysis expands the current knowledge by dissecting the contribution of a large number of transcript sequence-related features to differential protein abundance in *E. coli*. In addition to unraveling new determinants with significant impact on translation initiation, we also confirm the relevance of codon and amino acid usage to efficiency of translation.

Chapter 7

Rationally Designed Families of Orthogonal RNA Regulators of Translation

This chapter is based on the article:

Vivek K Mutalik, Lei Qi, **Joao C Guimaraes**, Julius B Lucks and Adam P Arkin. (2012). Rationally designed families of orthogonal RNA regulators of translation. *Nature Chemical Biology*, 8, 447-454.

7.1 Abstract

Our ability to routinely engineer genetic networks for applications is limited by the scarcity of highly specific and non-crossreacting (orthogonal) gene regulators with predictable behavior. Though regulatory antisense RNAs are attractive contenders for this purpose, quantitative understanding of their specificity to enable forward design has been limited. Here, we use rationally designed variants of the RNA-IN/OUT antisense RNA-mediated translation system from the insertion sequence IS10 to quantify >500 RNA-RNA interactions in *Escherichia coli* and integrate the data set with sequence-activity modeling to identify thermodynamic stability of the duplex and seed region as key determinants of specificity. Applying this model, we predict the performance of an additional ~2600 antisense-regulator pairs, forecast the possibility of larger families of orthogonal mutants, and forward engineer and experimentally validate two RNA pairs orthogonal to an existing group of five from the training dataset. We discuss the potential use of these regulators in next-generation synthetic biology applications.

7.2 Introduction

Precise control of gene expression is at the core of any genetic engineering dependent discipline. For most synthetic biology applications, it is necessary to express multiple genes at controlled levels possibly responsive to multiple internal and external signals. More specifically, the ability to rationally regulate the expression of multiple genes aids in the optimization of biosynthetic pathways for industrial chemical production (Keasling, 2008). Elements that allow controlled exploration of different pathway enzyme stoichiometries to maximize productivity or enable design of regulatory circuitry that balances enzyme activity to minimize toxic intermediates are valuable elements in the metabolic engineer's toolbox (Keasling, 2008; Pfleger et al., 2006). These parts are even more critical in emerging applications, such as cell-based therapies and active biomaterials that require more sophisticated regulatory circuitry implementing complex logic functions and memory (Ruder et al., 2011). It has been argued that a large compendium of diverse regulators that

are orthogonal (i.e., do not inadvertently crossreact), homogeneous (operate with similar kinetics, thermodynamics and other structural properties) and have predictable functionality are necessary to enable increasingly complex genetic circuit design (Lucks et al., 2008; Purnick and Weiss, 2009).

Though there has been some success in mining such regulatory elements from the vast library of function available in nature, most commonly promoters, each individual instance must be characterized itself, in its context, and needs further part and strain engineering to make it operate as desired in a given application (e.g., to match a desired dynamic range or not to interfere with host machinery). An alternative approach is to engineer part variants from a common ancestor that differ in designed ways from otherwise homogeneous operation. There has been an increasing effort to engineer such part libraries. Most efforts have focused on engineering promoters (Alper et al., 2005; Deuschle et al., 1986) and ribosome-binding sites (Salis et al., 2009) with desired activities. Models to predict the activities of new variants of these have exhibited some success (Barrick et al., 1994; Jonsson et al., 1993; Salis et al., 2009). Alternatively, there have been a number of efforts to engineer parts libraries that regulate different aspects of gene expression: transcription initiation with families of transcription factors (Zhan et al., 2010), transcriptional elongation via antisense RNA-regulated attenuators (Lucks et al., 2011) and unnatural amino acid regulated leader sequence expression (Liu et al., 2011); translation initiation using orthogonal ribosomes (Chubiz and Rao, 2008; Rackham and Chin, 2005), riboswitches (Dixon et al., 2010) and *trans*-regulatory RNAs (Isaacs et al., 2004); and mRNA stability via engineered ribozymes (Carothers et al., 2011; Win and Smolke, 2008), stability hairpins (Carrier and Keasling, 1999), RNase sites (Babiskin and Smolke, 2011; Smolke and Keasling, 2002) and protein sensing RNA devices (Culler et al., 2010; Saito et al., 2010). While foundational first steps, only few efforts have attempted to create models that link sequence to activity to allow forward design of new family members (Carothers et al., 2011; Chubiz and Rao, 2008; Salis et al., 2009).

Regulatory mechanisms that are mediated by RNA molecules, however, might be expected to be among the most amenable to such modeling. The apparent simplicity in their base pairing rules and modularity of structural components make them excellent substrates for design, and attractive contenders for a standard plat-

form for gene expression regulation in synthetic biological applications (Babiskin and Smolke, 2011; Benenson, 2009; Carothers et al., 2011; Isaacs et al., 2004; Pfleger et al., 2006). They also provide a powerful basis set of functions for gene expression engineering due to their ability to sense biomolecules, regulate transcriptional elongation, translation initiation, and mRNA degradation, and to act in *cis* arrayed on the same transcript or in *trans* to allow the construction of regulatory networks.

Albeit the versatility of RNA molecules has been recognized, there have been few studies on designing an entirely artificial (synthetic) riboregulator (cognate sense-antisense RNA) family or modifying a naturally occurring mechanism for use in gene silencing or activation purposes in *E. coli* (Engdahl et al., 1997; Isaacs et al., 2004; Lucks et al., 2011; Man et al., 2011; Nakashima and Tamura, 2009). In a seminal study (Isaacs et al., 2004), simple base pairing design rules were used to create a family of two orthogonal sense-antisense RNA partners that regulate translation. The interaction between cognate pairs showed a positive correlation with RNA-RNA binding free energies, implying a thermodynamically driven, one-step pathway to a stable duplex. Recently, a study from our lab reported a set of three orthogonal mutants of a natural antisense RNA-mediated transcriptional attenuation system and found a poor correlation between target repression and the thermodynamic stability of the duplex, suggesting that other factors besides thermodynamics may be at play (Lucks et al., 2011). These results are consistent with earlier observations that the specificity and efficiency shown by antisense RNA control systems may be explained by interactions between different structural motifs or modules and not solely by complementarity in base pairing (or hybridization free energies) (Thomason and Storz, 2010; Wagner et al., 2002). Notably, in both of these instances, the strategy for creating orthogonal mutants was based on the design features observed in other similar systems, and mutants were more or less handcrafted.

If scalable design of expression with RNA regulators is to succeed, quantitative sequence-function relationships that elucidate the mechanisms behind orthogonally acting RNA regulators and thus enable their rational design are needed. Sequence-activity relationships have proven useful in the annotation of genomes, making it possible to infer regulatory sites and other such features (Sahota and Stormo, 2010). Similar quantitative studies have provided mechanistic insights into the op-

eration of eukaryotic microRNAs (miRNAs) and small interfering RNAs in silencing target mRNAs and their off-target cross-talk (Rajewsky, 2006; Thomas et al., 2010). In these cases, statistical modeling has been used to mine large compendiums of experimental data to uncover specificity determinants for RNA-RNA interaction. These advances have aided the implementation of complex synthetic programs that can have an immense therapeutic value (Xie et al., 2011).

Here we present a minimal complexity, maximally predictive data driven sequence-activity model from quantitative assays of a mutant library derived from a well-known antisense RNA-mediated translational control system RNA-IN/OUT of IS10 in *E. coli* (Kittle et al., 1989). By model integration of hundreds of *in vivo* reporter assays, we identify the key determinants of antisense RNA interaction specificity. We then use the model to predict the performance of additional regulator pairs and experimentally validate these predictions by forward designing new orthogonal mutant pairs. Overall, this study demonstrates the utility of hypothesis driven library construction to parameterize predictive models of regulatory element function. These models yield insight into the critical mechanisms of the activity of these elements and provide computer-aided design tools for the production of new elements of the antisense RNA family.

7.3 Materials and methods

7.3.1 Strains, plasmids and growth conditions

E. coli strain Top10 (Invitrogen) was used for plasmid construction purposes and for fluorescence measurements. All strains were grown in Luria-Bertani (DIFCO) broth or MOPS EZ rich media (Teknova) at 37 °C with shaking. Media was supplemented with 100µg/ml carbenicillin and/or 34µg/ml chloramphenicol as required.

To retain the natural stoichiometry 1:20 of RNA-IN and RNA-OUT in our minimal assay system, we expressed antisense RNA-OUT from a high copy plasmid (ColE1, ~60 copies per cell), while sense RNA-IN was expressed from a low copy plasmid (SC101, ~5 copies per cell). Thus converting *cis* regulation of RNA-IN with RNA-OUT to *trans* allows easy and independent manipulation of both interactants.

We have also used stronger promoters transcribing both sense and antisense RNAs, and stronger translation initiation region for RNA-IN with a close to consensus Shine-Dalgarno (SD) region.

To test mutual orthogonality among all the five members of the orthogonal family, we translationally fused the sense regions to either *sfgfp*, hereafter *gfp*, or *mrfp1*, hereafter *rfp*, and repression was assayed in the presence of different antisense RNAs (cognate and noncognate) expressed from a different plasmid.

7.3.2 *In vivo* assays using plate reader and flow cytometer

Frozen Top10 cells carrying a RNA-IN fusion plasmid with plasmid expressing either RNA-OUT or nonsense RNA were grown overnight (~16 hours) in MOPS media with appropriate antibiotics at 37 °C/shaking. The following day overnight cultures were diluted 1:25 in to a fresh medium with appropriate antibiotics and 1mM isopropyl-β-D-thiogalactopyranoside (IPTG), and grown till cultures reached logarithmic growth phase. The measurements of OD_{600} and fluorescence (arbitrary fluorescence units; for super folder green fluorescent protein (GFP), excitation at 480 nm and emission at 510 nm; and for monomeric red fluorescent protein (RFP), excitation 560 nm and emission at 610nm) was performed in the Tecan Sapphire 2 spectrophotometer by diluting the log phase culture (0.5-0.6 OD_{600}) 1:2 in PBS buffer (pH 7.4). The measurement of fluorescence in the flow cytometer was performed by diluting the log phase culture 1:50 with PBS buffer (pH 7.4) containing 200 ug/mL Kanamycin. The plate was analyzed immediately using a Partec CyFlow Space flow cytometer with a RobbyWell 96-well plate auto sample loader. For each strain, a minimum of 50,000 cellular counts was collected using SSC as the cell trigger.

The target repression data is presented as percentage repression ± standard deviation or arbitrary fluorescence units/optical density ± standard deviation where applicable. The percentage repression from the plate reader data is calculated as:

$$\%\,Repression = \frac{Fluorescence_{null} - Fluorescence_{anti}}{Fluorescence_{null}} \times 100 \qquad \text{Eq. 7.1}$$

where $Fluorescence_{anti}$ and $Fluorescence_{null}$ represents the measured fluorescence in presence and absence of the antisense, respectively.

7.3.3 Sequence-activity model

To build a predictive model based on the sequence and thermodynamic features of sense, antisense and duplex RNA species and their corresponding percentage repression profiles, we used a partial least squares (PLS) regression (Wold et al., 2001). PLS is a method for relating two data matrices, response variable(s) Y to multiple predictors X, by a linear multivariate model. PLS finds components of X so that they approximate X and correlate with Y. This is the method of choice for handling multicollinearity when two or more predictor variables are highly correlated, and, hence, more robustly estimate the regression coefficients. The following equation shows the linear relationship between response variable and predictors:

$$y_i = \beta_0 + \beta_1 x_{j1} + \beta_2 x_{j2} + \cdots + \beta_t x_{jt} + \varepsilon_j \qquad \text{Eq. 7.2}$$

where y_i is a column vector containing the percentage repression data for a total of 529 data points for all 23 sense and antisense pairs; $x_{j1}, x_{j2}, \ldots, x_{jt}$ are the predictors in matrix X; $\beta_1, \beta_2, \ldots, \beta_t$ are the regression coefficients, β_0 is the regression constant and ε_j is the error term. The value of j ranges from 1 to 519 and the value of t ranges from 1 to 31. Thus, the matrix X contains computed normalized predictors and column vector Y contains the experimentally determined percentage of repression. The predictors 'ΔG' and 'seed_region_ΔG' showed nonlinear relationship when plotted against %Repression and, hence, were normalized using a sigmoidal function to meet the PLS requirement that all predictors need to have a linear behavior with respect to the response variable. Then, we used the Unscrambler X10 (CAMO software) for PLS model building, outlier detection and calculation of regression coefficients. To eliminate features with non-significant (P-value $>$ 0.05) regression weights, we used stepwise regression for iterative feature elimination procedure and find the two most significant (P-value $<$ 0.05) predictors for the final model. A total of eight candidates (out of the 529) with high residual y variance and high leverage were deemed outliers and were removed from the final model. To test whether the model overfitted the data, 10-fold cross-validation was performed. Finally, we used the model trained on the experimental data set to predict percentage repression of all 56 pair combinations.

7.4 Results

7.4.1 A minimal assay system for quantifying orthogonal mutants

Our translational regulators are derived from the copy number control element from the insertion sequence IS10, wherein an antisense RNA (RNA-OUT) inhibits transposase expression (Kittle et al., 1989). RNA-OUT base pairs to the translation initiation region of the transposase mRNA (RNA-IN) and represses translation both by preventing ribosome binding (Ma and Simons, 1990) and by promoting transcript degradation (Case et al., 1990) (Figure 7.1A). The 5' end of the unstructured sense RNA-IN is complementary to the top of the loop domain and to one entire side of the stable RNA-OUT hairpin (Figure 7.1B). Earlier studies have suggested that the 5 base pairs (bp) in the 5' end of RNA-IN and the loop domain of RNA-OUT determine the initiation of the RNA duplex formation (Jain, 1995; Kittle et al., 1989). The loop domain of RNA-OUT contains a pyrimidine-uracil-nucleotide-purine (YUNR) motif, and is predicted to promote specificity and rapid duplex formation with RNA-IN (Franch et al., 1999). The first three bp between RNA-IN and RNA-OUT are G-C pairs and the strength of hybridization free energy in this GC-rich region seems to be critical for effective antisense interaction and molecular specificity (Kittle et al., 1989). We reasoned, therefore, that these specificity-determining interactions could be manipulated to engineer families of orthogonal variants of the native system. We use the term orthogonal family to describe a group (more than two members) of sense and antisense mutants that specifically interact with their cognate partners and show minimal interaction with their noncognate counterparts.

To simplify the engineering of the RNA-IN/OUT system, we identified a minimized and slightly modified regulatory region that is sufficient for >85% repression of the target. To assess the performance of the RNA-IN/OUT system, we measured the percentage of repression of RNA-IN-mediated super folder green fluorescent protein (GFP) (Pedelacq et al., 2006) fluorescence (constitutively expressed from a low copy plasmid) in the presence of RNA-OUT (expressed from a high-copy plasmid) in *E. coli* TOP10 cells during exponential growth. We observed graded tuning

of target repression at different induction levels of antisense RNA and about 90% repression of GFP fluorescence when RNA-OUT was fully induced. As this performance agrees with previously reported data using much longer RNA-IN and RNA-OUT regions (used in conjunction with their endogenous promoters and a *lacZ* reporter system) (Jain, 1995; Kittle et al., 1989), we conclude that our minimized system retains the desired functionality.

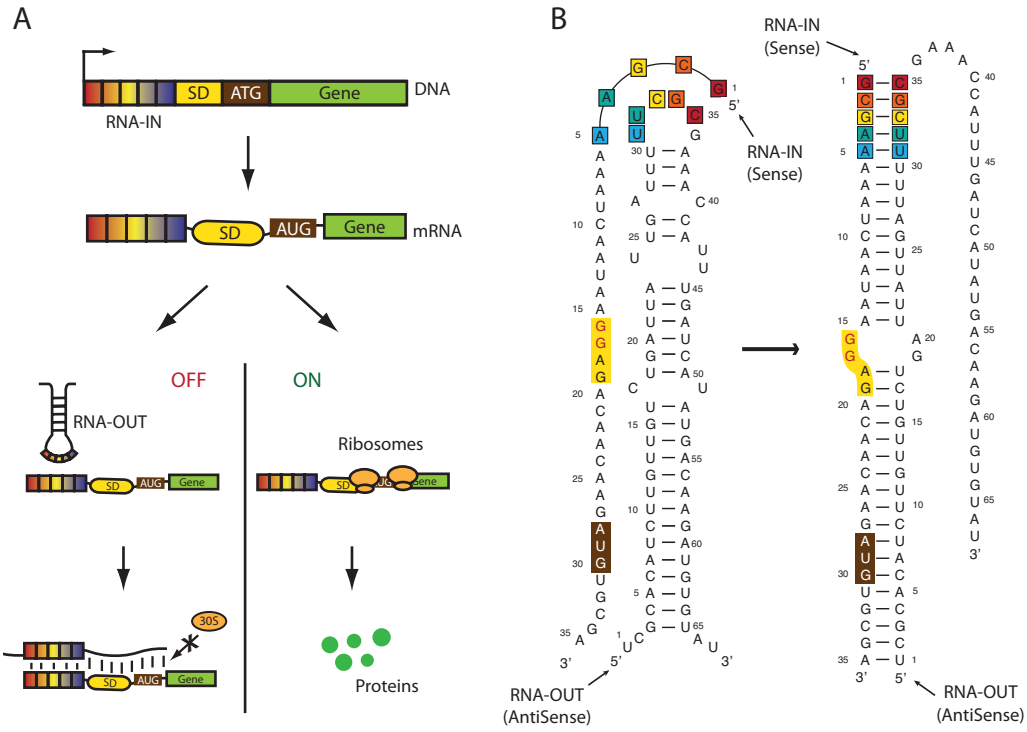


Figure 7.1 Schematic of the RNA-IN/OUT system
(A) Schematic of antisense RNA-regulated translation repression of the RNA-IN-*gfp* translational fusion. Antisense RNA (RNA-OUT, a stem-loop structure) binds specifically to RNA-IN-*gfp* mRNA, which blocks the Shine-Dalgarno (SD) sequence and start codon (AUG), thereby inhibiting its translation. Molecular recognition and initial base pairing between sense and antisense RNA is indicated by colored boxes. In the absence of RNA-OUT, translation of RNA-IN-*gfp* mRNA proceeds normally to produce GFP. (B) RNA-OUT—RNA-IN hybridization. The entire region of RNA-OUT used in the current work is not shown for clarity. Colored boxes indicate the core recognition region between interacting species. The mutated SD region in RNA-IN mRNA is marked as yellow, and the AUG start codon is colored brown.

7.4.2 Design of mutant library

To engineer mutually orthogonal sense-antisense pairs, we considered complementary mutations at the first five nucleotides (nt) in the 5' specificity region of RNA-IN and at the corresponding nucleotides in the loop of RNA-OUT (Figure

7.1B and Table 10). This led to a set of 32 variants in sense RNA-IN and antisense RNA-OUT. We reasoned that the possible number of orthogonal pairs could also be increased by inserting nucleotides within the recognition motif of this system, thereby "scaling up" the RNA-RNA interaction region. We therefore considered insertion of two extra nucleotides, AT, GC, TA or CG, between position 3 and 4 of RNA-IN (corresponding to complementary nucleotides at positions 32 and 33 of RNA-OUT; Figure 7.1B and Table 10). We also hypothesized that compensatory mutations in the first 3 bp of the interaction region in these scaled-up mutants would extend the number of orthogonal pairs and possibly improve repression efficiency. This resulted in 24 additional RNA-IN/OUT paired mutations for a total library size of 56 for each RNA element (Table 10). This number may be further increased by considering all possible combinations of (single, double and so on) nucleotide insertions with different combinations at the first 3 bp.

Earlier studies have suggested that the RNA-IN/OUT interaction is thermodynamically favored over strand exchange from the imperfectly base paired RNA-OUT hairpin to the perfectly base paired RNA-IN/OUT duplex (Jain, 1997); furthermore, our rationally constrained library of RNA-IN/OUT pairs is composed of mutants that have a 5 bp variable sequence region surrounded by a common flanking sequence (Table 10).

We thus assumed that the specificity of interaction and repression efficiency in our library of mutants could be explained, to a large extent, by differences in their hybridization free energies. The cognate pairs would be expected to have lower hybridization energy than noncognate pairs. To predict to the first order which pairs in our virtual library would show the highest specificity of interaction and the lowest cross-talk with other members, we estimated the hybridization free energies using UNAfold software (Markham and Zuker, 2008) for all 56 sense-antisense mutant pairs in the library (total 3,136 interactions). As expected, we found that the cognate sense-antisense partners (diagonal in Figure 7.2), showed far more stable hybrids compared to noncognate partners. To maximize the chance of mutual orthogonality, we selected 23 candidates from the total of 56 library members via a clustering procedure (Table 10). That only 5 out of the 23 RNA-OUT mutants conserve the YUNR motif also gave us a chance to test the importance of this motif in the functioning of the RNA-IN/OUT system.

Table 10 Selected sense and antisense mutants

The recognition motif (5' to 3') of 23 sense RNA-IN and 23 antisense RNA-OUT RNAs are shown with mutated or inserted positions from wild-type species denoted by single and double underlining, respectively. Total number of possible mutants from a particular base swap and the number of chosen mutants for experimental characterization are also shown.

Sense	Sequence	Antisense	Sequence	Nucleotide swaps	Tested/possible
S1	GCGAA	A1	UUCGC	wild-type	1/1
S3	GGGAA	A3	UUCCC	1	4/5
S4	GCCAA	A4	UUGGC		
S5	GCGUA	A5	UACGC		
S6	GCGAU	A6	AUCGC		
S7	CGGAA	A7	UUCCG	2	2/10
S8	CCCAA	A8	UUGGG		
S17	CGCAA	A17	UUGCG	3	6/10
S19	CGGAU	A19	AUCCG		
S21	CCCAU	A21	AUGGG		
S22	CCGUU	A22	AACGG		
S23	GGCUA	A23	UAGCC		
S26	GCCUU	A26	AAGGC		
S27	CGCUA	A27	UAGCG	4	5/5
S28	CGCAU	A28	AUGCG		
S29	CGGUU	A29	AACCG		
S30	CCCUU	A30	AAGGG		
S31	GGCUU	A31	AAGCC		
S32	CGCUU	A32	AAGCG	5	1/1
S34	GGGUAAA	A34	UUUACCC	Extra 2	4/24
S43	CCCAUAA	A43	UUAUGGG		
S49	CCCCGAA	A49	UUCGGGG		
S52	GGGGCAA	A52	UUGCCCC		

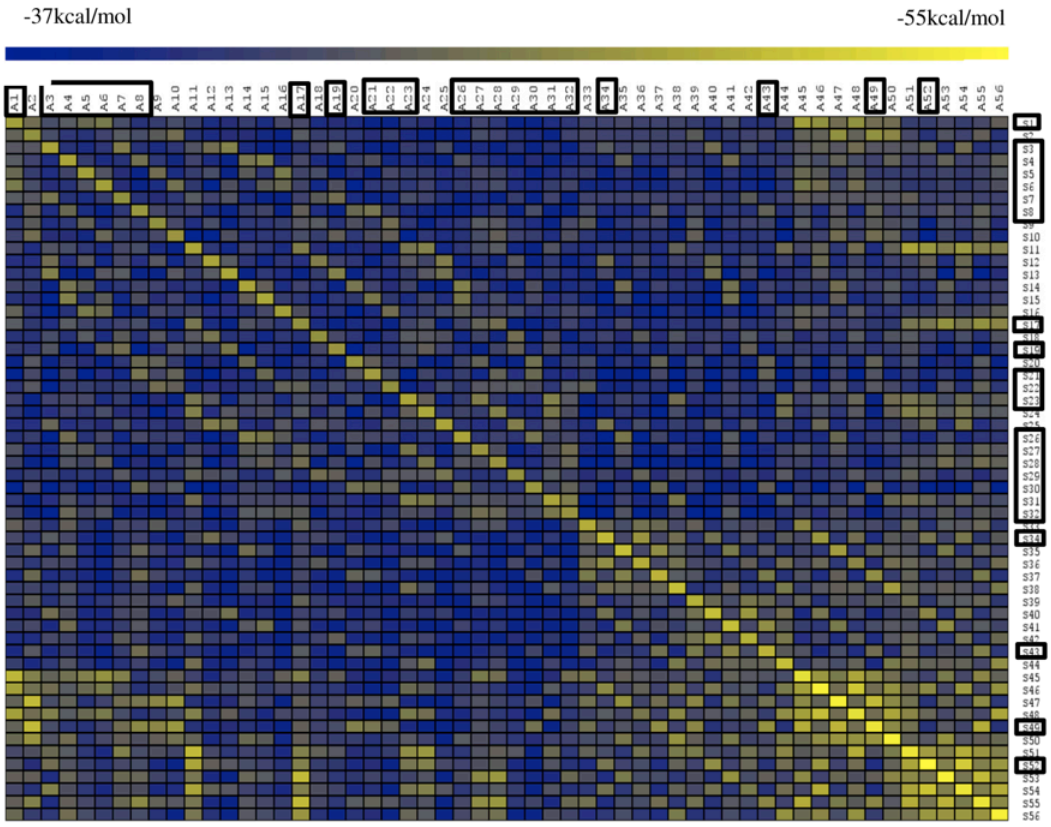


Figure 7.2 Heat map of the hybridization energy between the 56 different mutants
The interaction region (up to 37 nt) of 56 variants of RNA-IN and RNA-OUT was used to calculate the duplex hybridization free energy. The hybridization energy ranges from -55 to -37 kcal/mol. The cognate partners along the diagonal have predicted free energies closer to -50 kcal/mol and show more stable duplexes than those of the noncognate partners. Boxed mutants are the final chosen 23 pairs of sense and antisense RNA candidates. Orthogonal pairs are expected to have lower hybridization energy between the cognate pairs, but higher hybridization energy with noncognate pairs.

7.4.3 Measurement and analysis of the mutant library

We generated the 23 RNA-IN and RNA-OUT mutant constructs, and each of the 529 possible pairs was cotransformed on separate plasmids into *E. coli*. In addition, all RNA-IN plasmids were cotransformed with a nonsense RNA-OUT plasmid as a negative control. The performance of RNA-IN/OUT pairs was quantified by measuring the fluorescence during the exponential phase of each strain and percentage repression was calculated (Material and methods, and Eq. 7.1). The matrix of percentage repression for the 529 combinations of sense and antisense mutants is shown as a heat map in Figure 7.3. Most cognate sense-antisense pairs (along the diagonal on the heat map in Figure 7.3) show strong repression (>80%). Approxi-

mately 5% of all pairs achieve more than 70% repression, whereas about 75% of total pairs show less than 20% repression.

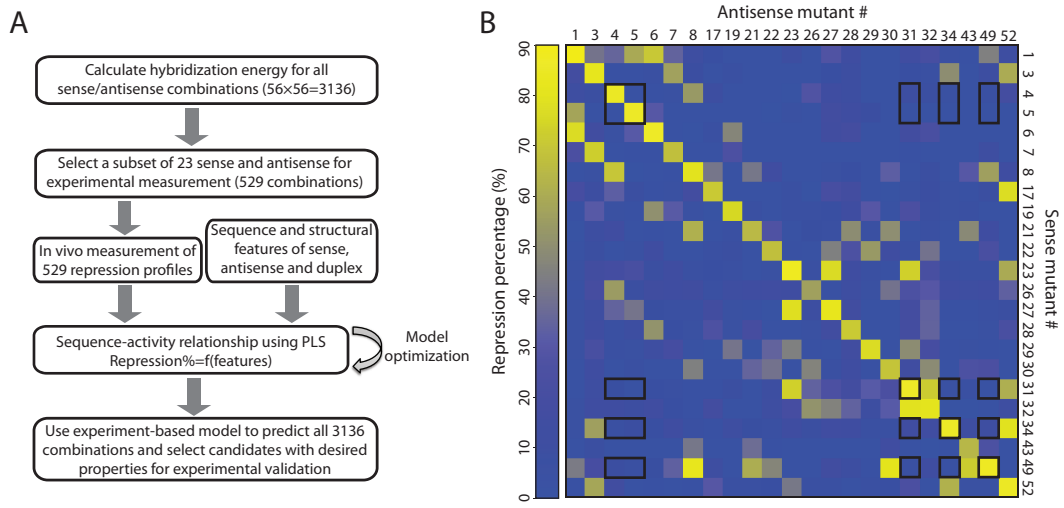


Figure 7.3 A rationally designed library for finding orthogonal pairs
(A) A quantitative framework to describe RNA-RNA interactions by integrating *in vivo* reporter assay data with sequence-activity modeling to identify specificity determinants and further enable forward engineering of orthogonal families of translation regulators. (B) Heat map of percentage repression profile of 23 RNA-IN mutants in presence of 23 antisense RNA-OUT mutants (total 529 data points). Cognate pairs are arranged diagonally and show maximum repression. Five mutually orthogonal pairs are shown as black boxes.

Overall, we observed a wide range of percentage target repression, ranging from negligible repression (<5%) to over 90% repression, and the change in dynamic range (ratio between target expression in absence and presence of antisense RNA) varied from zero to 10-fold. We also found many examples of a single antisense RNA repressing multiple sense targets (e.g., A52, Figure 7.3) and single sense targets being recognized by multiple antisense RNAs (e.g., S49, Figure 7.3). One notable result is that more than 70% of cognate pairs showed repression higher than 75% and did not have a YUNR motif in the antisense RNA species, indicating that the YUNR motif is not indispensable for the proper functioning of this system.

To determine how the energetics of the sense-antisense RNA interaction correlates with the experimental percentage repression, we plotted the calculated hybridization free energy for all 529 interactions against the experimental percentage repression (Figure 7.4A). We observed that hybridization of both cognate and non-cognate pairs with a free energy greater than -41 kcal/mol is not active in repression, whereas most duplexes with a free energy less than -46 kcal/mol showed

strongest repression (closer to 85%). These results indicate that there is a critical free energy threshold needed for the formation of a stable duplex and consequent effective repression of target mRNA. Similar results have been reported in eukaryotes for interaction of miRNAs with mRNA targets in HeLa cell lines (Doench and Sharp, 2004). We did not observe a strong correlation between target repression and the accessibility of the recognition motif (unfolding free energy of sense and antisense RNA). This indicates that the applied mutational strategy may interfere minimally with the structure of both interacting RNAs.

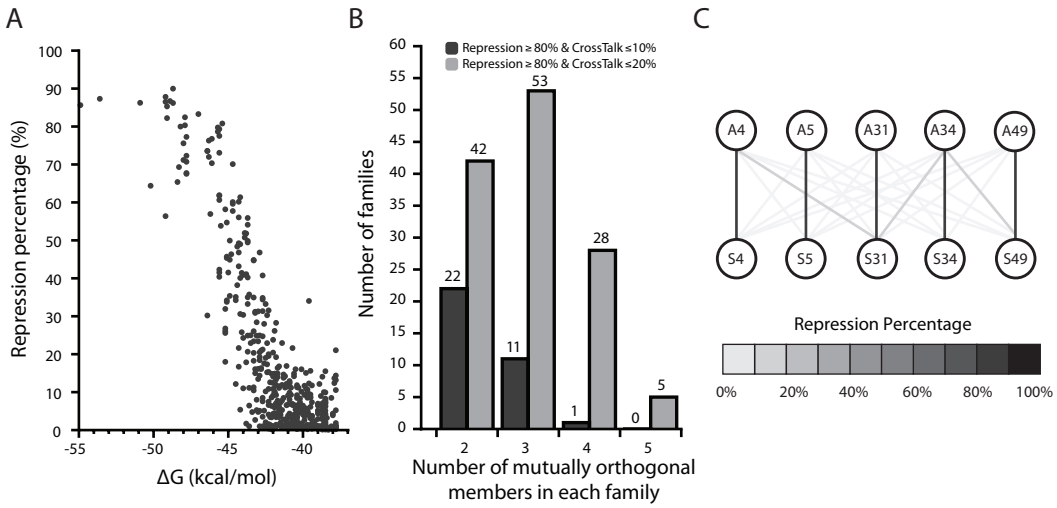


Figure 7.4 Repression characteristics and estimation of orthogonal pairs
(A) Scatter plot of calculated hybridization free energy (ΔG kcal/mol) as a function of experimental percentage repression. (B) Estimation of the number of families made up of two, three, four and five mutually orthogonal members at different thresholds of percentage repression and cross-talk. Two representative data sets are shown here: data for ≥80% repression and ≤10% cross-talk (dark grey bars) and data for ≥80% repression and ≤20% cross-talk (light grey bars). (C) The observed network of interactions between mutually orthogonal cognate and noncognate sense and antisense RNAs. The shading of links indicates the percentage repression (black for 100% repression and light gray for almost no repression).

7.4.4 Validating the orthogonality of mutant pairs

Using the experimentally determined percentage repression data to quantify target and nontarget specificity, we can identify groups or families of RNA-IN/OUT variants expected to function orthogonally when placed in the same cell. Further, identifying noncognate partners that show substantial cross-talk aids in determining base pairing features that impart promiscuity. Thus, the definition of mutual orthogonality depends on thresholds of repression and cross-reactivity percentages for

cognate and noncognate pairs, respectively, that we deem acceptable for a specific application.

The total number of observed mutants for different family sizes showing at least 80% repression with cognate and less than 10% or 20% cross-reactivity with other members is shown in Figure 7.4B. At an 80% repression threshold and 10% cross-reactivity, we have more than ten families of mutually orthogonal pairs and triplets and one family of four orthogonal mutants, whereas at 20% cross-reactivity, we have more than 20 families made up of two, three and four mutually orthogonal mutants and five families of five mutants.

To test whether the orthogonality and repression profile of these mutants was retained with a sequence-divergent gene of interest, we also fused the best five mutually orthogonal sense mutants (shown in Figure 7.4C) to the fluorescent protein RFP (52% sequence identity to GFP) and assayed them in the presence of corresponding antisense RNAs. The observed percentage repressions were quantitatively equivalent to that observed using GFP (Spearman's coefficient of rank correlation (*rho*) = 0.48, P-value = 0.02), demonstrating the modularity of the sense region and the efficiency of antisense RNA. To further demonstrate the mutual orthogonality among members of an orthogonal family in the same cell, we picked five sense-antisense pairs (shown in Figure 7.4C) and characterized every combination of two pairs in a single cell (Materials and methods). Here, the sense partner of each pair was translationally fused to either *gfp* or *rfp*, and repression was quantified in the presence of different combinations of the cognate antisense RNAs expressed from a different plasmid (Figure 7.5A). The results support the possibility that our library produced a large number of mutually orthogonal and modular regulatory variants that retain their specificity characteristics within the same cell (Figure 7.5B).

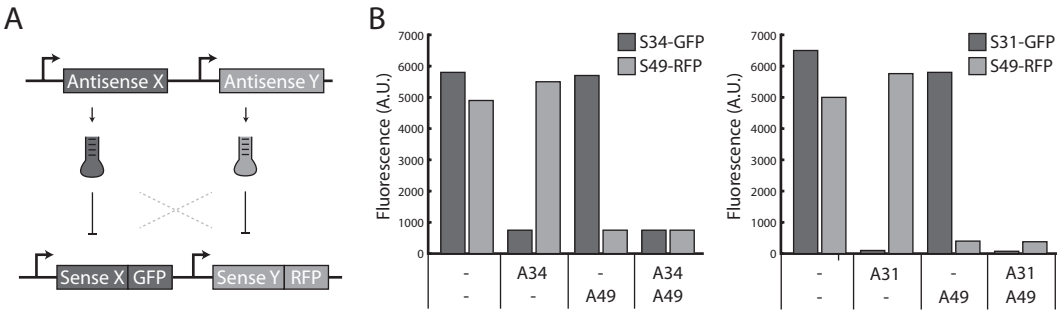


Figure 7.5 Two color experimental design to assess orthogonality in the same cell
(A) Expression regulation of two target RNAs fused to GFP and RFP by two orthogonal antisense RNA mutants. We performed all the ten combinations of sense and antisense for five orthogonal mutants. (B) Plots depict the mean fluorescence measured for two of these cultures expressing different combinations of sense regions fused to either GFP or RFP in presence and absence of the corresponding antisense molecules. No antisense control is shown by the symbol dash.

7.4.5 Sequence-activity relationship model

The correlation between the hybridization free energy and percentage repression suggests that free energy is a good, though not perfect, predictor of interaction specificity (Figure 7.4A). To find other features that determine the specificity of interaction between RNA-IN and RNA-OUT, we pursued modeling of the sequence-activity relationship for the *in vivo* experimental data set.

From the inspection of the predicted sense and antisense RNA secondary structures and the formed duplex, we shortlisted 31 possible features that might explain the observed patterns of repression (Table 11). Specifically, we compiled different sequence, thermodynamic and accessibility features to explain the base pairing specificity and consequent observed percentage of repression.

Table 11 Features considered for the sequence-activity model
We considered 31 sequence and structure related features to be used as predictors of the observed repression profile between the multiple sense and antisense RNA molecules

Feature	Description	Type	RNA species
ΔG	Hybridization energy of sense-antisense duplex	Floating point	Duplex
Sense-ΔG	Hybridization free energy of sense Minimum free energy (MFE) structure	Floating point	Sense
Antisense_ΔG	Hybridization free energy of antisense MFE structure	Floating point	Antisense
Seed_region_ΔG	Hybridization energy of duplex in 5 nt seed region	Floating point	Duplex
Seed_unpaired	Number of unpaired bases in duplex of 5 nt seed region	Integer	Duplex

Feature	Description	Type	RNA species
motif|AA|4	A4A5 (RNA-IN) – paired to (RNA-OUT) This feature is equal to 1 if the nucleotides AA (at position 4 and 5 in the RNA-IN molecule) are both paired in the sense-antisense duplex. Note that the pairing position on RNA-OUT is not fixed, that is A4A5 of RNA-IN can pair anywhere on RNA-OUT *not necessarily* at the same position as the WT pairs.	Boolean: 1 (paired) 0 (not paired)	Duplex
motif|AA|5	A5A6 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|UA|5	U5A6 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|UU|4	U4U5(RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|GC|1	G1C2 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|GC|2	G2C3 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|CG|1	C1G2 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|CG|2	C2G3 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|GG|1	G1G2 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|GG|2	G2G3 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|CC|1	C1C2 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|CC|2	C2C3 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|GA|3	G3A4 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|GU|3	G3U4 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|CA|3	C3A4 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motif|CU|3	C3U4 (RNA-IN) – paired to (RNA-OUT)	Boolean	Duplex
motifs_ΔG	Hybridization energy of dinucleotide motifs, considers just the free energy from bound nucleotides.	Float	Duplex
Interior_loops_number	Number of bulges in duplex	Integer	Duplex
Exterior_loop_ΔG	Hybridization free energy for the exterior loop of sense-antisense duplex	Float	Duplex
s_paired_6	Paired status of nucleotide position 6 of sense MFE structure (change in structure when nuclotide 1-5 are mutated)	Boolean	Sense
s_exterior_loop_ΔG	Hybridization free energy for the exterior loop of sense MFE structure	Float	Sense
s_exterior_loop_ss	Number of bases single stranded from sense MFE structure	Integer	Sense
s_seed_region_ΔG	Hybridization free energy in 5 bp seed region of sense MFE structure	Float	Sense
s_seed_unpaired	Number of bases unpaired in 5 bp seed region of sense MFE structure	Integer	Sense
as_seed_region_ΔG	Hybridization free energy in 5bp seed region of antisense MFE structure	Float	Antisense
as_seed_unpaired	Number of bases unpaired in 5bp seed region of antisense MFE structure	Integer	Antisense

Then, to select the most important feature explaining the repression data, we applied partial least squares (PLS) regression with stepwise feature selection and out-

lier detection (Materials and methods). The analysis, after detecting and discarding 8 outlier interactions (out of 529 interactions), identified just two features that explained the 86% variation in the data after 10-fold cross-validation: the hybridization energies of the entire 37-bp interaction region and that of the 5-bp seed region (Figure 7.6A). The model suggests that the initial nucleation event at the GC-rich 5-bp seed region and the subsequent helix progression is thermodynamically driven and determines the efficient repression of the target mRNA. These results recapitulate early studies that pointed out the importance of the 5-bp interaction region in determining the copy-number control performance of the RNA-IN/OUT system (Kittle et al., 1989).

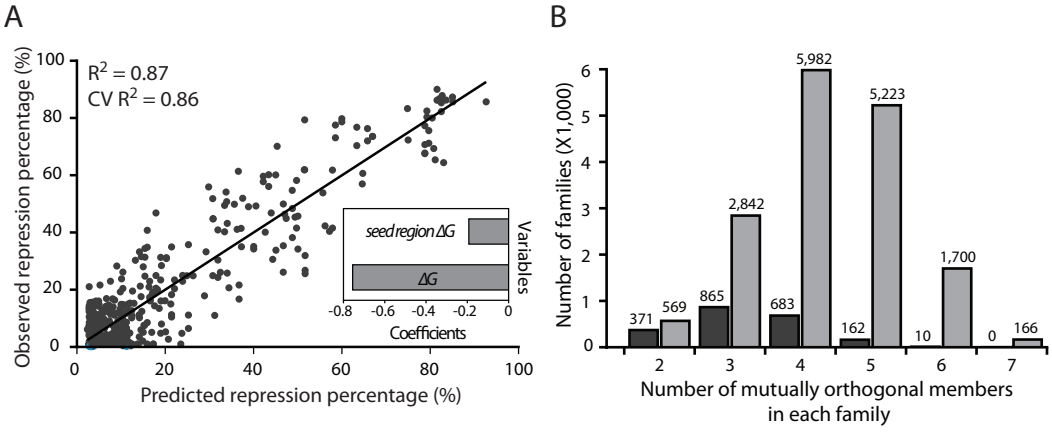


Figure 7.6 The sequence-activity PLS model
(A) The PLS model results are shown as a scatter plot of predicted versus observed percentage repression. The coefficient of determination (a measure of the quality of the model, $R^2 = 0.87$, $P < 0.0001$) using a training set of 521 interactions is shown along with the 10-fold cross-validated model performance (a measure of the predictive ability of the model, cross-validated (CV) $R^2 = 0.86$) Inset, weighted regression coefficients for the final two predictors. These include ΔG, hybridization energy of the sense-antisense RNA duplex; and seed region ΔG, hybridization energy of duplex in seed region. (B) The estimated number of families with two to seven mutually orthogonal members in the computationally predicted repression profile of the 56 pairs. Two representative data sets are shown here for repression ≥80%: data for ≤10% cross-talk (dark grey bars) and ≤20% cross-talk (light grey bars).

The unexplained 14% variance in the repression data may be due to other features or factors that are not included in this work, such as the *in vivo* concentrations of interacting RNAs, which influence the final repression efficiency (Jain, 1997). The eight outlier RNA interactions removed from the training model, which had high residual variance and high leverage, did not have any obvious properties that explained the deviant behavior. We speculate that the peculiarity of the structure or

the *in vivo* stability of duplexes of these outliers may be the reason for their unpredictable performance. More detailed biochemical studies are needed to pursue these hypotheses. However, the model, trained on the remaining 521 pairs, has sufficient explanatory power to support the design of new pairs.

7.4.6 Validation of model predictions

To validate the predictive capability of the model and forward engineer new orthogonal mutants, we used the model to predict the percentage repression for all interactions between the 56 mutant pairs initially considered (Figure 7.3A). This yielded a total of 3,136 percentage repression predictions, including the 529 experimentally tested pairs. We estimated the total possible number of mutually orthogonal pairs in the 56 RNA-IN and RNA-OUT variants from these predictions for different family sizes at different thresholds of percentage repression and cross-talk (Figure 7.6B). At 80% threshold percentage repression and 10% cross-reactivity, we have more than 300 families of mutually orthogonal pairs, triplets and quadruplets; more than 150 families of 5 mutants; and 10 families of 6 orthogonal mutants. At 20% cross-reactivity, we have more than 1,000 families made up of 3, 4, 5 and 6 mutually orthogonal mutants and more than 150 families of 7 mutants.

To experimentally validate a subset of these predictions, we forward engineered two sense and antisense RNA pairs (mutant 13 and mutant 40) predicted to have a desired strong threshold of percentage repression and minimal cross-talk with the family of five orthogonal pairs discovered in our initial screening. Specifically, mutant 40 was predicted to form a new family of five orthogonal pairs, whereas mutant 13 was predicted to expand the current family with a sixth orthogonal member (Figure 7.7). We characterized these four forward-engineered mutants in the presence of their cognate and noncognate partners. As predicted, sense and antisense mutants 13 and 40 yielded an expanded and a new family made up of six and five mutually orthogonal mutants, respectively (Figure 7.7).

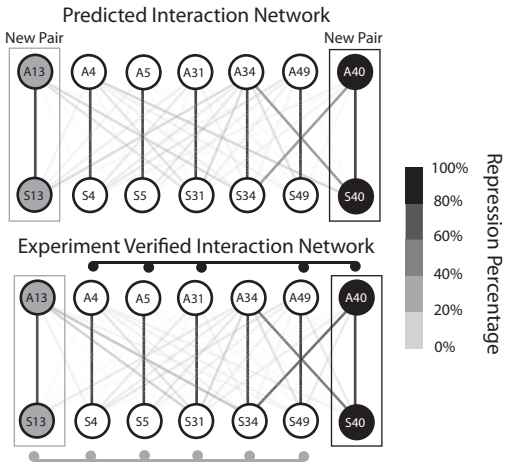


Figure 7.7 Experimental validation of model predictions
Predicted (top) and experimentally verified (bottom) network of interaction between mutually orthogonal sense and antisense RNAs (mutants 13 and 40 in addition to initial mutants 4, 5, 31, 34 and 49). The shading of links indicates the percentage repression.

7.5 Discussion

The translational controllers we developed here, and the model that enables the design of effective and orthogonal regulator pairs, provide an effective platform for RNA-based regulation of translation. The platform allowed the discovery of a large number of families of orthogonal regulators that could, theoretically, be used in the same cell. The model enabled the engineering of new families of five and six mutually orthogonal translational regulators showing consistent and predictable performance. To our knowledge, these are the largest orthogonal families constructed from a single regulatory mechanism based on a predictive model (in contrast to commonly used screening approaches). In addition, we forecast the possibility of many more mutually orthogonal families that may have up to 7 elements (Figure 7.6B). We also demonstrate that the specificity of interaction is preserved when extra nucleotides are inserted in the core interaction region, thereby opening up an avenue to allow the search of additional regulators in the enlarged sequence space.

Analysis of the library also provides insight into the mechanisms of specificity and activity in the RNA-IN/OUT system. For example, earlier studies showed that single complementary mutations at the third and fourth nucleotide at the 5' end of RNA-IN (and at the corresponding nucleotides in the loop region of RNA-OUT) alter the sequence specificity of the antisense pairing reaction with their wild-type

counterparts (Kittle et al., 1989). In the present work, in addition to recapitulating these results, we observe that combinatorial complementary nucleotide swaps (1-5 nucleotides) in any of the five first nucleotides alter the specificity of interaction with each other and with their wild-type partners, indicating the extreme importance of the initial base pairing region in the stable duplex formation.

Further, our results demonstrate that the ubiquitously observed YUNR motif in antisense RNA systems seems to be nonessential for retaining the specificity and efficiency of the interaction in the RNA-IN/OUT system. In a few specific cases, antisense mutants with no YUNR motif show as good repression as the wild-type (e.g., A6, A5, A23, A31, A52; Figure 7.3B). A few mutants that do not show a strong interaction with their cognate partners (e.g., A7, A21, A22, A30 and A43; Figure 7.3B) also do not have a YUNR motif, and further characterization is needed to understand whether concentration or structural features influence their performance. These results indicate that the YUNR motif as a principal design feature may not be universally applicable to all synthetic RNA regulators, and more elaborate studies are needed to show whether there are any structural dependencies for this motif to function as a key determinant of interaction.

Our model selection procedure suggests that the hybridization energy of the entire duplex and that of the 5-bp duplex seed region are the important features determining the sense-antisense RNA interaction specificity. The importance of the seed region suggests that the initial nucleating events and interaction strength of this region in the propagation of the base pairing reaction are critical to specificity and repression. Earlier work suggested that only the first 3 bp of the RNA-IN/OUT duplex and their free energy were important for this nucleating event (Jain, 1997). The seed region has also been found to be a key determinant of activity in eukaryotic miRNA studies (Rajewsky, 2006; Thomas et al., 2010) and, more recently, in bacterial small RNAs (Papenfort et al., 2010). The role of the longer duplex hybridization energy in explaining the majority of *in vivo* repression data suggests that initiation and propagation of the nucleating event is crucial for efficient repression. Overall, the good agreement between the RNA hybridization thermodynamics and the behavior of the RNA-IN/OUT system enables the design of new variants that function equivalently and orthogonally to the wild-type element. These insights can be readily used for designing structurally analogous but sequence-independent anti-

sense RNAs and corresponding sense RNAs. Further, the experimental data-driven modeling approach presented here can also be applied to other RNA species (that may rely on similar or different modes of interaction).

By describing the thresholds of repression and cross-reactivity percentages for cognate and noncognate pairs, respectively, we can quantify mutual orthogonality and begin to define specification for parts appropriate for a particular application. Such an analysis is of immense value when these RNA regulators are integrated into synthetic genetic circuits and are required to function in a predictable way to avoid unwanted cross-talk with other parts in the circuit and host. Some applications (e.g., a circuit regulating cell death) demand a very precise and specific regulation of part components, whereas in some applications this requirement may not be that stringent. It is likely, for example, that as the number of orthogonal pairs to be used together in a cell increases, there will be an increasingly stringent requirement for a low cross-talk as the apparent nonspecific repression of a sense target might be affected by the sum of the concentrations of all cross-talking antisense molecules.

Nonetheless, the combination of orthogonal and promiscuous variants (i.e., a single antisense RNA repressing multiple sense targets and vice-versa) can also be extremely useful either in designing synthetic circuits as signal propagation and/or signal integration modules (Beisel and Storz, 2010; Engdahl et al., 1997) or for building a hierarchy of interlinked functional modules to understand the network structure and function (Beisel and Storz, 2010). The use of well-characterized parts to engineer interlinked genetic networks reminiscent of natural systems can yield insights into organization principles of circuit design and, more generally, how evolution may have shaped these complex regulatory networks.

We envision that both the large families of orthogonal regulators and the approach used to discover them will also be important for a variety of next-generation synthetic biology applications (Purnick and Weiss, 2009). This includes the use of orthogonal translation repressors to perform computations in bacteria (Benenson, 2009) or modulate protein abundance to perturb and improve pathway balance (Keasling, 2008; Pfleger et al., 2006; Smolke and Keasling, 2002). Even though the design specification for these various applications may not be very well defined or realized, having a compendium of well-characterized parts to meet the diverse speci-

fication requirements saves a lot of resources when compared to *ad hoc* approaches. The characterization and standardization of genetic parts is particularly important because the real potential of using RNA based components in genetic circuits derives from their inherent designability, homogeneity and scalability when compared to protein regulators. Therefore, we believe that the use of RNA components described in this work in combination with the many available transcriptional and post-transcriptional regulatory components and other recently developed orthogonal systems (An and Chin, 2009; Culler et al., 2010; Dixon et al., 2010; Isaacs et al., 2004; Lucks et al., 2011; Rackham and Chin, 2005; Win and Smolke, 2008; Zhan et al., 2010) should provide an unsurpassed platform for tailoring synthetic genetic circuits.

Chapter 8

Conclusion

The work presented in this thesis tackles some of the main challenges in achieving predictable control of gene expression. It also focuses on the development of an analysis and modeling framework that aids interrogating and engineering natural systems. The main conclusions of the resulting work are:

- The employment of combinatorial experimental designs coupled with well establish statistical frameworks, such as ANOVA, provides a powerful methodology to quantify the main effects of multiple factors under study, as well as the interactions between factors, on a given response variable. To demonstrate and validate this approach, we studied the effect of transcription and translation initiation elements on the expression levels of two different fluorescent proteins. Specifically, we quantified the average effect of each element to the steady-state protein levels and how such effect is perturbed as an element is reused across different contexts. We propose this latter characteristic to be used as a proxy of the genetic element's quality. The most significant deviant behavior resulted from interactions between translation initiation elements and downstream gene sequences. This observation immediately points out limitations of current gene expression control platforms and motivates the future development of more robust design strategies.
- The assessment of the interactions between transcription and translation initiation elements, and different gene sequences emphasized the need to devel-

op a genetic platform delivering predictable control of gene expression. To that end, we defined transcription initiation elements with a defined transcriptional start site and translation initiation elements using an overlapping genetic motif that ensured reliable expression of sequence-divergent genes across 1,000-fold observed dynamic range. This new platform should enable engineers to express any gene of interest within twofold relative to target expression with ~93% reliability, without having to worry that neighboring DNA will interfere or even silence the selected genetic elements (parts).

- Gene expression control systems are inherently complex due to the myriad of different regulatory signals that can be present in relatively small segments of DNA. Though an integrative analysis of the multiple sequence signals is necessary to fully understand how expression levels are tuned, the scattering of sequence analysis tools hinders the ability to automate this process. We developed D-Tailor, a software that precisely addresses this challenge by establishing a framework to integrate multiple sequence analysis tools into a single platform that provides the easy multi-factorial interrogation of DNA/RNA sequences. Its modular architecture further ensures that incremental improvements can be made to the established pipeline (e.g., definition of evaluation modules for new factors). Anticipating future design capabilities afforded by the increasing throughput of synthesis and measurement technologies, D-Tailor incorporates the ability to controllably design sequences over a wide range of user-defined parameters. Such designs will be essential to elucidate and quantify the effect of the different sequence determinants, as well as their interactions, on any observed phenotype of interest.

- An integrative approach enabled the assessment and quantification of the more than 100 determinants on the expression level of the endogenous genes in *E. coli*. Specifically, we demonstrated that a minimal set of 16 non-redundant determinants explained 66% of total variation of protein abundance across the entire genome. We show that mRNA transcript level is the main predictor of protein levels (53% of total variability), and that codon and amino acid usage biases are also responsible for a fraction of the variation (12%). The latter observation is particularly important because it has

recently been under scientific debate whether or not determinants of translation elongation actually affect steady-state protein levels. We verified that translation initiation modestly affects the variation of protein abundances (1%). We also identified a new sequence determinant of the translation initiation complex that significantly affects protein levels and we speculate that it might be related to the efficient dissociation of the 16S:SD complex and the subsequent start of the elongation step. Lastly, the general correlation found between transcription and translation efficiencies suggests that highly abundant mRNAs are efficiently translated, presumably to avoid the depletion of free ribosomes in the cell.

- A detailed understanding of natural systems controlling gene expression can be used, in some cases, to guide the rational engineering of the synthetic ones. We demonstrate the validity of this methodology using a natural RNA antisense system controlling translation efficiency. Specifically, we used a synthetic biology approach to generate multiple genetic variants of the wild-type system and study what were the main determinants of specificity of this regulatory system. The experimental characterization of a very large number of variants enabled the parameterization of a statistical model explaining the *in vivo* performance of the different mutants with an accuracy of 87%. The model not only enabled the identification of the sequence determinants impacting the wild-type system, but also uncovered design rules to generate new variants that are functionally homogeneous to the natural system and have predictable performance.

The work presented in this thesis describes a set of computational models that enabled the dissection of multiple determinants of protein levels in both natural and synthetic genetic circuits. Though the models developed are appropriate for the genetic circuits under study, we should be aware of their putative limited extendibility to other systems. For example, while the first order linear model was able to accurately capture the relationship between the multiple factors studied (transcription and translational initiation elements, gene sequences and temperature) and the experimentally measured protein levels (Chapters 3 and 4), we cannot guarantee that such simplistic mapping will hold true for other genetic or physical factors.

Likewise, the two sequence features identified as major determinants of the RNA antisense specificity studied in Chapter 7 may not provide identical predictive power in similar RNA silencing systems. Nonetheless, the incremental availability of models predicting biological function from a set of relatively easy computable input variables (Chapters 6 and 7) will ultimately lead to the development of a whole-cell model that simulates and integrates all cellular components and dynamics.

The gene expression platform described in Chapter 4 should enable engineers to express any gene of interest within twofold relative to target expression with ~93% reliability. Though this represents a significant improvement in predictability for synthetic genetic circuits, future efforts dissecting the main determinants of the overlapping genetic motif performance used by this platform will be essential to improve this reliability even further. It will also be of great value to populate the expression platform developed here with novel and more sophisticated functional elements other than the existing constitutive transcription and translation initiation capabilities.

Finally, the existence of software enabling the automated multi-factorial interrogation of DNA sequences (Chapter 5) provides an efficient tool to explore natural genomes and, when coupled with appropriate computational models, may help to elucidate the relevance of the multiple sequence determinants on the observed phenotype of interest (Chapters 6 and 7). Despite the modest diversity of sequence features evaluators implemented in our first version of D-Tailor, its modular architecture should engage researchers/developers with different areas of expertise to contribute for incrementally improved versions implementing a wider range of functionalities. In addition, the development of more advanced algorithms, such as the adoption of methods implementing populations of solutions or even the use of simulated annealing for the optimization stage, may further improve the efficiency of the sequence design algorithm. Such contributions will greatly improve D-Tailor's capacity to controllably and efficiently design artificial DNA segments across multiple parameters of interest for subsequent deployment into living organisms as afforded by the astonishing advances in high-throughput synthesis and screening technologies.

Bibliography

Aitken, C.E., Petrov, A., and Puglisi, J.D. (2010). Single ribosome dynamics and the mechanism of translation. *Annu Rev Biophys* **39**, 491-513.

Akashi, H. (1994). Synonymous codon usage in Drosophila melanogaster: natural selection and translational accuracy. *Genetics* **136**, 927-935.

Akashi, H. (2003). Translational selection and yeast proteome evolution. *Genetics* **164**, 1291-1303.

Akashi, H., and Gojobori, T. (2002). Metabolic efficiency and amino acid composition in the proteomes of Escherichia coli and Bacillus subtilis. *Proc Natl Acad Sci U S A* **99**, 3695-3700.

Allert, M., Cox, J.C., and Hellinga, H.W. (2010). Multifactorial determinants of protein expression in prokaryotic open reading frames. *J Mol Biol* **402**, 905-918.

Alper, H., Fischer, C., Nevoigt, E., and Stephanopoulos, G. (2005). Tuning genetic control through promoter engineering. *Proc Natl Acad Sci U S A* **102**, 12678-12683.

An, W., and Chin, J.W. (2009). Synthesis of orthogonal transcription-translation networks. *Proc Natl Acad Sci U S A* **106**, 8477-8482.

Anderson, J.C., Clarke, E.J., Arkin, A.P., and Voigt, C.A. (2006). Environmentally controlled invasion of cancer cells by engineered bacteria. *J Mol Biol* **355**, 619-627.

Anderson, J.C., Voigt, C.A., and Arkin, A.P. (2007). Environmental signal integration by a modular AND gate. *Mol Syst Biol* **3**, 133.

Andersson, S.G., and Kurland, C.G. (1990). Codon preferences in free-living microorganisms. *Microbiol Rev* **54**, 198-210.

Arkin, A., Ross, J., and McAdams, H.H. (1998). Stochastic kinetic analysis of developmental pathway bifurcation in phage lambda-infected Escherichia coli cells. *Genetics* **149**, 1633-1648.

Babiskin, A.H., and Smolke, C.D. (2011). A synthetic library of RNA control modules for predictable tuning of gene expression in yeast. *Mol Syst Biol* **7**, 471.

Babitzke, P., and Romeo, T. (2007). CsrB sRNA family: sequestration of RNA-binding regulatory proteins. *Curr Opin Microbiol* **10**, 156-163.

Bailey, T.L., Boden, M., Buske, F.A., Frith, M., Grant, C.E., Clementi, L., Ren, J., Li, W.W., and Noble, W.S. (2009). MEME SUITE: tools for motif discovery and searching. *Nucleic Acids Res* **37**, W202-208.

Barrick, D., Villanueba, K., Childs, J., Kalil, R., Schneider, T.D., Lawrence, C.E., Gold, L., and Stormo, G.D. (1994). Quantitative analysis of ribosome binding sites in E.coli. *Nucleic Acids Res* **22**, 1287-1295.

Bashor, C.J., Horwitz, A.A., Peisajovich, S.G., and Lim, W.A. (2010). Rewiring cells: synthetic biology as a tool to interrogate the organizational principles of living systems. *Annu Rev Biophys* **39**, 515-537.

Beisel, C.L., and Storz, G. (2010). Base pairing small RNAs and their roles in global regulatory networks. *FEMS Microbiol Rev* **34**, 866-882.

Belle, A., Tanay, A., Bitincka, L., Shamir, R., and O'Shea, E.K. (2006). Quantification of protein half-lives in the budding yeast proteome. *Proc Natl Acad Sci U S A* **103**, 13004-13009.

Benenson, Y. (2009). RNA-based computation in live cells. *Curr Opin Biotechnol* **20**, 471-478.

Benjamini, Y., and Hochberg, Y. (1995). Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing. *Journal of the Royal Statistical Society Series B (Methodological)* **57**, 289-300.

Beyer, A., Hollunder, J., Nasheuer, H.P., and Wilhelm, T. (2004). Post-transcriptional expression regulation in the yeast Saccharomyces cerevisiae on a genomic scale. *Mol Cell Proteomics* **3**, 1083-1092.

Boni, I.V. (2006). Diverse molecular mechanisms for translation initiation in prokaryotes. *Mol Biol (Mosk)* **40**, 658-668.

Bonnet, J., Subsoontorn, P., and Endy, D. (2012). Rewritable digital data storage in live cells via engineered control of recombination directionality. *Proc Natl Acad Sci U S A* **109**, 8884-8889.

Brantl, S. (2002). Antisense-RNA regulation and RNA interference. *Biochim Biophys Acta* **1575**, 15-25.

Brockmann, R., Beyer, A., Heinisch, J.J., and Wilhelm, T. (2007). Posttranscriptional expression regulation: what determines translation rates? *PLoS Comput Biol* **3**, e57.

Bulmer, M. (1988). Codon usage and intragenic position. *J Theor Biol* **133**, 67-71.

Bulmer, M. (1991). The selection-mutation-drift theory of synonymous codon usage. *Genetics* **129**, 897-907.

Cambray, G., and Mazel, D. (2008). Synonymous genes explore different evolutionary landscapes. *PLoS Genet* **4**, e1000256.

Cambray, G., Mutalik, V.K., and Arkin, A.P. (2011). Toward rational design of bacterial genomes. *Curr Opin Microbiol* **14**, 624-630.

Campbell, R.E., Tour, O., Palmer, A.E., Steinbach, P.A., Baird, G.S., Zacharias, D.A., and Tsien, R.Y. (2002). A monomeric red fluorescent protein. *Proc Natl Acad Sci U S A* **99**, 7877-7882.

Cannarozzi, G., Schraudolph, N.N., Faty, M., von Rohr, P., Friberg, M.T., Roth, A.C., Gonnet, P., Gonnet, G., and Barral, Y. (2010). A role for codon order in translation dynamics. *Cell* **141**, 355-367.

Canton, B., Labno, A., and Endy, D. (2008). Refinement and standardization of synthetic biological parts and devices. *Nat Biotechnol* **26**, 787-793.

Cardinale, S., and Arkin, A.P. (2012). Contextualizing context for synthetic biology--identifying causes of failure of synthetic biological systems. *Biotechnol J* **7**, 856-866.

Carothers, J.M., Goler, J.A., Juminaga, D., and Keasling, J.D. (2011). Model-driven engineering of RNA devices to quantitatively program gene expression. *Science* **334**, 1716-1719.

Carr, P.A., and Church, G.M. (2009). Genome engineering. *Nat Biotechnol* **27**, 1151-1162.

Carrier, T.A., and Keasling, J.D. (1999). Library of synthetic 5' secondary structures to manipulate mRNA stability in Escherichia coli. *Biotechnol Prog* **15**, 58-64.

Case, C.C., Simons, E.L., and Simons, R.W. (1990). The IS10 transposase mRNA is destabilized during antisense RNA control. *Embo J* **9**, 1259-1266.

Chan, L.Y., Kosuri, S., and Endy, D. (2005). Refactoring bacteriophage T7. *Mol Syst Biol* **1**, 2005 0018.

Chang, J.T., Green, C.B., and Wolf, R.E., Jr. (1995). Inhibition of translation initiation on Escherichia coli gnd mRNA by formation of a long-range secondary structure involving the ribosome binding site and the internal complementary sequence. *J Bacteriol* **177**, 6560-6567.

Chen, H., Bjerknes, M., Kumar, R., and Jay, E. (1994). Determination of the optimal aligned spacing between the Shine-Dalgarno sequence and the translation initiation codon of Escherichia coli mRNAs. *Nucleic Acids Res* **22**, 4953-4957.

Chen, S., Zhang, A., Blyn, L.B., and Storz, G. (2004). MicC, a second small-RNA regulator of Omp protein expression in Escherichia coli. *J Bacteriol* **186**, 6689-6697.

Chen, Y.Y., Jensen, M.C., and Smolke, C.D. (2010). Genetic control of mammalian T-cell proliferation with synthetic RNA regulatory systems. *Proc Natl Acad Sci U S A* **107**, 8531-8536.

Cho, K.O., and Yanofsky, C. (1988). Sequence changes preceding a Shine-Dalgarno region influence trpE mRNA translation and decay. *J Mol Biol* **204**, 51-60.

Chubiz, L.M., and Rao, C.V. (2008). Computational design of orthogonal ribosomes. *Nucleic Acids Res* **36**, 4038-4046.

Cobb, R.E., Sun, N., and Zhao, H. (2012). Directed evolution as a powerful synthetic biology tool. *Methods*.

Cortez, P. (2010). Data Mining with Neural Networks and Support Vector Machines using the R/rminer Tool. *Advances in Data Mining -- Applications and Theoretical Aspects, 10th Industrial Conference on Data Mining*, 572--583.

Cox, R.S., 3rd, Surette, M.G., and Elowitz, M.B. (2007). Programming gene expression with combinatorial promoters. *Mol Syst Biol* **3**, 145.

Crick, F. (1970). Central dogma of molecular biology. *Nature* **227**, 561-563.

Culler, S.J., Hoff, K.G., and Smolke, C.D. (2010). Reprogramming cellular behavior with RNA controllers responsive to endogenous proteins. *Science* **330**, 1251-1255.

Danino, T., Mondragon-Palomino, O., Tsimring, L., and Hasty, J. (2010). A synchronized quorum of genetic clocks. *Nature* **463**, 326-330.

Das, A., and Yanofsky, C. (1984). A Ribosome Binding-Site Sequence Is Necessary for Efficient Expression of the Distal Gene of a Translationally-Coupled Gene Pair. *Nucleic Acids Res* **12**, 4757-4768.

Davis, J.H., Rubin, A.J., and Sauer, R.T. (2011). Design, construction and characterization of a set of insulated bacterial promoters. *Nucleic Acids Res* **39**, 1131-1141.

de Smit, M.H., and van Duin, J. (1990). Secondary structure of the ribosome binding site determines translational efficiency: a quantitative analysis. *Proc Natl Acad Sci U S A* **87**, 7668-7672.

de Smit, M.H., and van Duin, J. (1994). Control of translation by mRNA secondary structure in Escherichia coli. A quantitative analysis of literature data. *J Mol Biol* **244**, 144-150.

Deana, A., and Belasco, J.G. (2005). Lost in translation: the influence of ribosomes on bacterial mRNA decay. *Genes Dev* **19**, 2526-2533.

Deuschle, U., Kammerer, W., Gentz, R., and Bujard, H. (1986). Promoters of Escherichia coli: a hierarchy of in vivo strength indicates alternate structures. *Embo J* **5**, 2987-2994.

Dixon, N., Duncan, J.N., Geerlings, T., Dunstan, M.S., McCarthy, J.E., Leys, D., and Micklefield, J. (2010). Reengineering orthogonally selective riboswitches. *Proc Natl Acad Sci U S A* **107**, 2830-2835.

Doench, J.G., and Sharp, P.A. (2004). Specificity of microRNA target selection in translational repression. *Genes Dev* **18**, 504-511.

Dong, H., Nilsson, L., and Kurland, C.G. (1996). Co-variation of tRNA abundance and codon usage in Escherichia coli at different growth rates. *J Mol Biol* **260**, 649-663.

Dubendorff, J.W., and Studier, F.W. (1991). Controlling basal expression in an inducible T7 expression system by blocking the target T7 promoter with lac repressor. *J Mol Biol* **219**, 45-59.

Ellinger, T., Behnke, D., Knaus, R., Bujard, H., and Gralla, J.D. (1994). Context-dependent effects of upstream A-tracts. Stimulation or inhibition of Escherichia coli promoter function. *J Mol Biol* **239**, 466-475.

Ellis, T., Adie, T., and Baldwin, G.S. (2011). DNA assembly for synthetic biology: from parts to pathways and beyond. *Integr Biol (Camb)* **3**, 109-118.

Ellis, T., Wang, X., and Collins, J.J. (2009). Diversity-based, model-guided construction of synthetic gene networks with predicted functions. *Nat Biotechnol* **27**, 465-471.

Elowitz, M.B., and Leibler, S. (2000). A synthetic oscillatory network of transcriptional regulators. *Nature* **403**, 335-338.

Elowitz, M.B., Levine, A.J., Siggia, E.D., and Swain, P.S. (2002). Stochastic gene expression in a single cell. *Science* **297**, 1183-1186.

Endy, D. (2005). Foundations for engineering biology. *Nature* **438**, 449-453.

Engdahl, H.M., Hjalt, T.A., and Wagner, E.G. (1997). A two unit antisense RNA cassette test system for silencing of target genes. *Nucleic Acids Res* **25**, 3218-3227.

Engler, C., Kandzia, R., and Marillonnet, S. (2008). A one pot, one step, precision cloning method with high throughput capability. *PLoS One* **3**, e3647.

Eyre-Walker, A. (1996). Synonymous codon bias is related to gene length in Escherichia coli: selection for translational accuracy? *Mol Biol Evol* **13**, 864-872.

Franch, T., Petersen, M., Wagner, E.G., Jacobsen, J.P., and Gerdes, K. (1999). Antisense RNA regulation in prokaryotes: rapid RNA/RNA interaction facilitated by a general U-turn loop structure. *J Mol Biol* **294**, 1115-1125.

Gardner, T.S., Cantor, C.R., and Collins, J.J. (2000). Construction of a genetic toggle switch in Escherichia coli. *Nature* **403**, 339-342.

Gaspar, P., Moura, G., Santos, M.A., and Oliveira, J.L. (2013). mRNA secondary structure optimization using a correlated stem-loop prediction. *Nucleic Acids Res*.

Gaspar, P., Oliveira, J.L., Frommlet, J., Santos, M.A., and Moura, G. (2012). EuGene: maximizing synthetic gene design for heterologous expression. *Bioinformatics* **28**, 2683-2684.

Giardine, B., Riemer, C., Hardison, R.C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., *et al.* (2005). Galaxy: a platform for interactive large-scale genome analysis. *Genome Res* **15**, 1451-1455.

Gibson, D.G., Glass, J.I., Lartigue, C., Noskov, V.N., Chuang, R.Y., Algire, M.A., Benders, G.A., Montague, M.G., Ma, L., Moodie, M.M., *et al.* (2010). Creation of a bacterial cell controlled by a chemically synthesized genome. *Science* **329**, 52-56.

Gingold, H., and Pilpel, Y. (2011). Determinants of translation efficiency and accuracy. *Mol Syst Biol* **7**, 481.

Gonzalez, B., Schwimmer, L.J., Fuller, R.P., Ye, Y., Asawapornmongkol, L., and Barbas, C.F., 3rd (2010). Modular system for the construction of zinc-finger libraries and proteins. *Nat Protoc* **5**, 791-810.

Gottesman, S., and Storz, G. (2011). Bacterial small RNA regulators: versatile roles and rapidly evolving variations. *Cold Spring Harb Perspect Biol* **3**.

Greenbaum, D., Colangelo, C., Williams, K., and Gerstein, M. (2003). Comparing protein abundance and mRNA expression levels on a genomic scale. *Genome Biol* **4**, 117.

Greenbaum, D., Jansen, R., and Gerstein, M. (2002). Analysis of mRNA expression and protein abundance data: an approach for the comparison of the enrichment of features in the cellular population of proteins and transcripts. *Bioinformatics* **18**, 585-596.

Gulvanessian, H., and Holicky, M. (2005). Eurocodes: using reliability analysis to combine action effects. *P I Civil Eng-Str B* **158**, 243-252.

Gustafsson, C., Govindarajan, S., and Minshull, J. (2004). Codon bias and heterologous protein expression. *Trends Biotechnol* **22**, 346-353.

Hall, M.N., Gabay, J., Debarbouille, M., and Schwartz, M. (1982). A role for mRNA secondary structure in the control of translation initiation. *Nature* **295**, 616-618.

Hertz, G.Z., and Stormo, G.D. (1999). Identifying DNA and protein patterns with statistically significant alignments of multiple sequences. *Bioinformatics* **15**, 563-577.

Hook-Barnard, I.G., and Hinton, D.M. (2007). Transcription initiation by mix and match elements: flexibility for polymerase binding to bacterial promoters. *Gene Regul Syst Bio* **1**, 275-293.

Iizuka, R., Yamagishi-Shirasaki, M., and Funatsu, T. (2011). Kinetic study of de novo chromophore maturation of fluorescent proteins. *Anal Biochem* **414**, 173-178.

Ikemura, T. (1985). Codon usage and tRNA content in unicellular and multicellular organisms. *Mol Biol Evol* **2**, 13-34.

Ingolia, N.T., Ghaemmaghami, S., Newman, J.R., and Weissman, J.S. (2009). Genome-wide analysis in vivo of translation with nucleotide resolution using ribosome profiling. *Science* **324**, 218-223.

Iost, I., Guillerez, J., and Dreyfus, M. (1992). Bacteriophage T7 RNA polymerase travels far ahead of ribosomes in vivo. *J Bacteriol* **174**, 619-622.

Isaacs, F.J., Dwyer, D.J., Ding, C., Pervouchine, D.D., Cantor, C.R., and Collins, J.J. (2004). Engineered riboregulators enable post-transcriptional control of gene expression. *Nat Biotechnol* **22**, 841-847.

Jain, C. (1995). IS10 antisense control in vivo is affected by mutations throughout the region of complementarity between the interacting RNAs. *J Mol Biol* **246**, 585-594.

Jain, C. (1997). Models for pairing of IS10 encoded antisense RNAs in vivo. *J Theor Biol* **186**, 431-439.

Jonsson, J., Norberg, T., Carlsson, L., Gustafsson, C., and Wold, S. (1993). Quantitative sequence-activity models (QSAM)--tools for sequence design. *Nucleic Acids Res* **21**, 733-739.

Keasling, J.D. (2008). Synthetic biology for synthetic chemistry. *ACS Chem Biol* **3**, 64-76.

Keasling, J.D. (2010). Manufacturing molecules through metabolic engineering. *Science* **330**, 1355-1358.

Khalil, A.S., and Collins, J.J. (2010). Synthetic biology: applications come of age. *Nat Rev Genet* **11**, 367-379.

Kingsford, C.L., Ayanbule, K., and Salzberg, S.L. (2007). Rapid, accurate, computational discovery of Rho-independent transcription terminators illuminates their relationship to DNA uptake. *Genome Biol* **8**, R22.

Kireeva, M.L., and Kashlev, M. (2009). Mechanism of sequence-specific pausing of bacterial RNA polymerase. *Proc Natl Acad Sci U S A* **106**, 8900-8905.

Kittle, J.D., Simons, R.W., Lee, J., and Kleckner, N. (1989). Insertion sequence IS10 anti-sense pairing initiates by an interaction between the 5' end of the target RNA and a loop in the anti-sense RNA. *J Mol Biol* **210**, 561-572.

Kittleson, J.T., Wu, G.C., and Anderson, J.C. (2012). Successes and failures in modular genetic engineering. *Curr Opin Chem Biol* **16**, 329-336.

Klumpp, S., Zhang, Z., and Hwa, T. (2009). Growth rate-dependent global effects on gene expression in bacteria. *Cell* **139**, 1366-1375.

Kozak, M. (1999). Initiation of translation in prokaryotes and eukaryotes. *Gene* **234**, 187-208.

Kudla, G., Murray, A.W., Tollervey, D., and Plotkin, J.B. (2009). Coding-sequence determinants of gene expression in Escherichia coli. *Science* **324**, 255-258.

Kwok, R. (2010). Five hard truths for synthetic biology. *Nature* **463**, 288-290.

Lee, K., Holland-Staley, C.A., and Cunningham, P.R. (1996). Genetic analysis of the Shine-Dalgarno interaction: selection of alternative functional mRNA-rRNA combinations. *Rna* **2**, 1270-1285.

Lee, S.Y., Kim, H.U., Park, J.H., Park, J.M., and Kim, T.Y. (2009). Metabolic engineering of microorganisms: general strategies and drug production. *Drug Discov Today* **14**, 78-88.

Lee, T.S., Krupa, R.A., Zhang, F., Hajimorad, M., Holtz, W.J., Prasad, N., Lee, S.K., and Keasling, J.D. (2011). BglBrick vectors and datasheets: A synthetic biology platform for gene expression. *J Biol Eng* **5**, 12.

Leveau, J.H., and Lindow, S.E. (2001). Predictive and interpretive simulation of green fluorescent protein expression in reporter bacteria. *J Bacteriol* **183**, 6752-6762.

Levine, E., and Hwa, T. (2008). Small RNAs establish gene expression thresholds. *Curr Opin Microbiol* **11**, 574-579.

Levine, E., Zhang, Z., Kuhlman, T., and Hwa, T. (2007). Quantitative characteristics of gene regulation by small RNA. *PLoS Biol* **5**, e229.

Liljenstrom, H., and von Heijne, G. (1987). Translation rate modification by preferential codon usage: intragenic position effects. *J Theor Biol* **124**, 43-55.

Liu, C.C., Qi, L., Yanofsky, C., and Arkin, A.P. (2011). Regulation of transcription by unnatural amino acids. *Nat Biotechnol* **29**, 164-168.

Lo, K., Hahne, F., Brinkman, R.R., and Gottardo, R. (2009). flowClust: a Bioconductor package for automated gating of flow cytometry data. *BMC Bioinformatics* **10**, 145.

Lou, C., Stanton, B., Chen, Y.J., Munsky, B., and Voigt, C.A. (2012). Ribozyme-based insulator parts buffer synthetic circuits from genetic context. *Nat Biotechnol* **30**, 1137-1142.

Lu, P., Vogel, C., Wang, R., Yao, X., and Marcotte, E.M. (2007). Absolute protein expression profiling estimates the relative contributions of transcriptional and translational regulation. *Nat Biotechnol* **25**, 117-124.

Lu, T.K., Khalil, A.S., and Collins, J.J. (2009). Next-generation synthetic gene networks. *Nat Biotechnol* **27**, 1139-1150.

Lucks, J.B., Qi, L., Mutalik, V.K., Wang, D., and Arkin, A.P. (2011). Versatile RNA-sensing transcriptional regulators for engineering genetic networks. *Proc Natl Acad Sci U S A* **108**, 8617-8622.

Lucks, J.B., Qi, L., Whitaker, W.R., and Arkin, A.P. (2008). Toward scalable parts families for predictable design of biological circuits. *Curr Opin Microbiol* **11**, 567-573.

Lutz, R., and Bujard, H. (1997). Independent and tight regulation of transcriptional units in Escherichia coli via the LacR/O, the TetR/O and AraC/I1-I2 regulatory elements. *Nucleic Acids Res* **25**, 1203-1210.

Ma, C., and Simons, R.W. (1990). The IS10 antisense RNA blocks ribosome binding at the transposase translation initiation site. *Embo J* **9**, 1267-1274.

Maier, T., Guell, M., and Serrano, L. (2009). Correlation of mRNA and protein in complex biological samples. *FEBS Lett* **583**, 3966-3973.

Maier, T., Schmidt, A., Guell, M., Kuhner, S., Gavin, A.C., Aebersold, R., and Serrano, L. (2011). Quantification of mRNA and protein and integration with protein turnover in a bacterium. *Mol Syst Biol* **7**, 511.

Makoff, A.J., and Smallwood, A.E. (1990). The use of two-cistron constructions in improving the expression of a heterologous gene in E. coli. *Nucleic Acids Res* **18**, 1711-1718.

Man, S., Cheng, R., Miao, C., Gong, Q., Gu, Y., Lu, X., Han, F., and Yu, W. (2011). Artificial trans-encoded small non-coding RNAs specifically silence the selected gene expression in bacteria. *Nucleic Acids Res* **39**, e50.

Mandal, M., and Breaker, R.R. (2004). Gene regulation by riboswitches. *Nat Rev Mol Cell Biol* **5**, 451-463.

Markham, N.R., and Zuker, M. (2008). UNAFold: software for nucleic acid folding and hybridization. *Methods Mol Biol* **453**, 3-31.

Masuda, T., Saito, N., Tomita, M., and Ishihama, Y. (2009). Unbiased quantitation of Escherichia coli membrane proteome using phase transfer surfactants. *Mol Cell Proteomics* **8**, 2770-2777.

Mathews, D.H., and Turner, D.H. (2006). Prediction of RNA secondary structure by free energy minimization. *Curr Opin Struct Biol* **16**, 270-278.

McAdams, H.H., and Arkin, A. (1997). Stochastic mechanisms in gene expression. *Proc Natl Acad Sci U S A* **94**, 814-819.

McDowell, J.C., Roberts, J.W., Jin, D.J., and Gross, C. (1994). Determination of intrinsic transcription termination efficiency by RNA polymerase elongation rate. *Science* **266**, 822-825.

Mendez-Perez, D., Gunasekaran, S., Orler, V.J., and Pfleger, B.F. (2012). A translation-coupling DNA cassette for monitoring protein translation in Escherichia coli. *Metab Eng* **14**, 298-305.

Mertens, N., Remaut, E., and Fiers, W. (1995). Tight transcriptional control mechanism ensures stable high-level expression from T7 promoter-based expression plasmids. *Biotechnology (N Y)* **13**, 175-179.

Mevik, B.-H., and Wehrens, R. (2007). The pls Package: Principal Component and Partial Least Squares Regression in R. *Journal of Statistical Software* **18**, 1-24.

Miller, R.G. (1986). Beyond ANOVA, basics of applied statistics (New York, Wiley).

Mutalik, V.K., Guimaraes, J.C., Cambray, G., Mai, Q.-A., Christoffersen, M.J., Martin, L., Yu, A., Lam, C., Rodrigues, C., Bennett, G.*, et al.* (2013). Quantitative Estimation of Activity and Quality for Collections of Functional Genetic Elements. *Nature Methods*.

Mutalik, V.K., Nonaka, G., Ades, S.E., Rhodius, V.A., and Gross, C.A. (2009). Promoter strength properties of the complete sigma E regulon of Escherichia coli and Salmonella enterica. *J Bacteriol* **191**, 7279-7287.

Mutalik, V.K., Qi, L., Guimaraes, J.C., Lucks, J.B., and Arkin, A.P. (2012). Rationally designed families of orthogonal RNA regulators of translation. *Nat Chem Biol* **8**, 447-454.

Nakashima, N., and Tamura, T. (2009). Conditional gene silencing of multiple genes with antisense RNAs and generation of a mutator strain of Escherichia coli. *Nucleic Acids Res* **37**, e103.

Nie, L., Wu, G., and Zhang, W. (2006). Correlation of mRNA expression and protein abundance affected by multiple sequence features related to translational efficiency in Desulfovibrio vulgaris: a quantitative analysis. *Genetics* **174**, 2229-2243.

Oppenheim, D.S., and Yanofsky, C. (1980). Translational coupling during expression of the tryptophan operon of Escherichia coli. *Genetics* **95**, 785-795.

Ozbudak, E.M., Thattai, M., Kurtser, I., Grossman, A.D., and van Oudenaarden, A. (2002). Regulation of noise in the expression of a single gene. *Nat Genet* **31**, 69-73.

Papenfort, K., Bouvier, M., Mika, F., Sharma, C.M., and Vogel, J. (2010). Evidence for an autonomous 5' target recognition domain in an Hfq-associated small RNA. *Proc Natl Acad Sci U S A* **107**, 20435-20440.

Pedelacq, J.D., Cabantous, S., Tran, T., Terwilliger, T.C., and Waldo, G.S. (2006). Engineering and characterization of a superfolder green fluorescent protein. *Nat Biotechnol* **24**, 79-88.

Peters, J.M., Mooney, R.A., Kuan, P.F., Rowland, J.L., Keles, S., and Landick, R. (2009). Rho directs widespread termination of intragenic and stable RNA transcription. *Proc Natl Acad Sci U S A* **106**, 15406-15411.

Pfleger, B.F., Pitera, D.J., Smolke, C.D., and Keasling, J.D. (2006). Combinatorial engineering of intergenic regions in operons tunes expression of multiple genes. *Nat Biotechnol* **24**, 1027-1032.

Plotkin, J.B., and Kudla, G. (2011). Synonymous but not the same: the causes and consequences of codon bias. *Nat Rev Genet* **12**, 32-42.

Purnick, P.E., and Weiss, R. (2009). The second wave of synthetic biology: from modules to systems. *Nat Rev Mol Cell Biol* **10**, 410-422.

Qi, L., Haurwitz, R.E., Shao, W., Doudna, J.A., and Arkin, A.P. (2012). RNA processing enables predictable programming of gene expression. *Nat Biotechnol* **30**, 1002-1006.

Qu, X., Wen, J.D., Lancaster, L., Noller, H.F., Bustamante, C., and Tinoco, I., Jr. (2011). The ribosome uses two active mechanisms to unwind messenger RNA during translation. *Nature* **475**, 118-121.

Quan, J., Saaem, I., Tang, N., Ma, S., Negre, N., Gong, H., White, K.P., and Tian, J. (2011). Parallel on-chip gene synthesis and application to optimization of protein expression. *Nat Biotechnol* **29**, 449-452.

Rackham, O., and Chin, J.W. (2005). A network of orthogonal ribosome x mRNA pairs. *Nat Chem Biol* **1**, 159-166.

Raj, A., and van Oudenaarden, A. (2008). Nature, nurture, or chance: stochastic gene expression and its consequences. *Cell* **135**, 216-226.

Rajewsky, N. (2006). microRNA target predictions in animals. *Nat Genet* **38 Suppl**, S8-13.

Rhodius, V.A., and Mutalik, V.K. (2010). Predicting strength and function for promoters of the Escherichia coli alternative sigma factor, sigmaE. *Proc Natl Acad Sci U S A* **107**, 2854-2859.

Ringquist, S., Shinedling, S., Barrick, D., Green, L., Binkley, J., Stormo, G.D., and Gold, L. (1992). Translation initiation in Escherichia coli: sequences within the ribosome-binding site. *Mol Microbiol* **6**, 1219-1229.

Ro, D.K., Paradise, E.M., Ouellet, M., Fisher, K.J., Newman, K.L., Ndungu, J.M., Ho, K.A., Eachus, R.A., Ham, T.S., Kirby, J.*, et al.* (2006). Production of the antimalarial drug precursor artemisinic acid in engineered yeast. *Nature* **440**, 940-943.

Rosenfeld, N., Young, J.W., Alon, U., Swain, P.S., and Elowitz, M.B. (2007). Accurate prediction of gene feedback circuit behavior from component properties. *Mol Syst Biol* **3**, 143.

Ruder, W.C., Lu, T., and Collins, J.J. (2011). Synthetic biology moving into the clinic. *Science* **333**, 1248-1252.

Saeidi, N., Wong, C.K., Lo, T.M., Nguyen, H.X., Ling, H., Leong, S.S., Poh, C.L., and Chang, M.W. (2011). Engineering microbes to sense and eradicate Pseudomonas aeruginosa, a human pathogen. *Mol Syst Biol* **7**, 521.

Sahota, G., and Stormo, G.D. (2010). Novel sequence-based method for identifying transcription factor binding sites in prokaryotic genomes. *Bioinformatics* **26**, 2672-2677.

Saito, H., Kobayashi, T., Hara, T., Fujita, Y., Hayashi, K., Furushima, R., and Inoue, T. (2010). Synthetic translational regulation by an L7Ae-kink-turn RNP switch. *Nat Chem Biol* **6**, 71-78.

Salis, H.M., Mirsky, E.A., and Voigt, C.A. (2009). Automated design of synthetic ribosome binding sites to control protein expression. *Nat Biotechnol* **27**, 946-950.

Schoner, B.E., Belagaje, R.M., and Schoner, R.G. (1986). Translation of a synthetic two-cistron mRNA in Escherichia coli. *Proc Natl Acad Sci U S A* **83**, 8506-8510.

Schumperli, D., Mckenney, K., Sobieski, D.A., and Rosenberg, M. (1982). Translational Coupling at an Intercistronic Boundary of the Escherichia-Coli Galactose Operon. *Cell* **30**, 865-871.

Schwanhausser, B., Busse, D., Li, N., Dittmar, G., Schuchhardt, J., Wolf, J., Chen, W., and Selbach, M. (2011). Global quantification of mammalian gene expression control. *Nature* **473**, 337-342.

Serganov, A., and Patel, D.J. (2007). Ribozymes, riboswitches and beyond: regulation of gene expression without proteins. *Nat Rev Genet* **8**, 776-790.

Sharp, P.M., and Li, W.H. (1987). The codon Adaptation Index--a measure of directional synonymous codon usage bias, and its potential applications. *Nucleic Acids Res* **15**, 1281-1295.

Shcherbakov, D.V., and Garber, M.B. (2000). Overlapping genes in bacterial and bacteriophage genomes. *Mol Biol (Mosk)* **34**, 572-583.

Shimada, T., Makinoshima, H., Ogawa, Y., Miki, T., Maeda, M., and Ishihama, A. (2004). Classification and strength measurement of stationary-phase promoters by use of a newly developed promoter cloning vector. *J Bacteriol* **186**, 7112-7122.

Shine, J., and Dalgarno, L. (1975). Determinant of cistron specificity in bacterial ribosomes. *Nature* **254**, 34-38.

Shultzaberger, R.K., Bucheimer, R.E., Rudd, K.E., and Schneider, T.D. (2001). Anatomy of Escherichia coli ribosome binding sites. *J Mol Biol* **313**, 215-228.

Sinha, J., Reyes, S.J., and Gallivan, J.P. (2010). Reprogramming bacteria to seek and destroy an herbicide. *Nat Chem Biol* **6**, 464-470.

Smolke, C.D. (2009). Building outside of the box: iGEM and the BioBricks Foundation. *Nat Biotechnol* **27**, 1099-1102.

Smolke, C.D., and Keasling, J.D. (2002). Effect of copy number and mRNA processing and stabilization on transcript and protein levels from an engineered dual-gene operon. *Biotechnol Bioeng* **78**, 412-424.

Spanjaard, R.A., and van Duin, J. (1989). Translational reinitiation in the presence and absence of a Shine and Dalgarno sequence. *Nucleic Acids Res* **17**, 5501-5507.

Sprinzak, D., and Elowitz, M.B. (2005). Reconstruction of genetic circuits. *Nature* **438**, 443-448.

Steitz, J.A. (1969). Polypeptide chain initiation: nucleotide sequences of the three ribosomal binding sites in bacteriophage R17 RNA. *Nature* **224**, 957-964.

Stougaard, P., Molin, S., and Nordstrom, K. (1981). RNAs involved in copy-number control and incompatibility of plasmid R1. *Proc Natl Acad Sci U S A* **78**, 6008-6012.

Stricker, J., Cookson, S., Bennett, M.R., Mather, W.H., Tsimring, L.S., and Hasty, J. (2008). A fast, robust and tunable synthetic gene oscillator. *Nature* **456**, 516-519.

Studer, S.M., and Joseph, S. (2006). Unfolding of mRNA secondary structure by the bacterial translation initiation complex. *Mol Cell* **22**, 105-115.

Swain, P.S., Elowitz, M.B., and Siggia, E.D. (2002). Intrinsic and extrinsic contributions to stochasticity in gene expression. *Proc Natl Acad Sci U S A* **99**, 12795-12800.

Takyar, S., Hickerson, R.P., and Noller, H.F. (2005). mRNA helicase activity of the ribosome. *Cell* **120**, 49-58.

Taniguchi, Y., Choi, P.J., Li, G.W., Chen, H., Babu, M., Hearn, J., Emili, A., and Xie, X.S. (2010). Quantifying E. coli proteome and transcriptome with single-molecule sensitivity in single cells. *Science* **329**, 533-538.

Team, R.D.C. (2011). R: A Language and Environment for Statistical Computing. *R Foundation for Statistical Computing*.

Temme, K., Zhao, D., and Voigt, C.A. (2012). Refactoring the nitrogen fixation gene cluster from Klebsiella oxytoca. *Proc Natl Acad Sci U S A* **109**, 7085-7090.

Thattai, M., and van Oudenaarden, A. (2001). Intrinsic noise in gene regulatory networks. *Proc Natl Acad Sci U S A* **98**, 8614-8619.

Thomas, M., Lieberman, J., and Lal, A. (2010). Desperately seeking microRNA targets. *Nat Struct Mol Biol* **17**, 1169-1174.

Thomas-Chollier, M., Defrance, M., Medina-Rivera, A., Sand, O., Herrmann, C., Thieffry, D., and van Helden, J. (2011). RSAT 2011: regulatory sequence analysis tools. *Nucleic Acids Res* **39**, W86-91.

Thomason, M.K., and Storz, G. (2010). Bacterial antisense RNAs: how many are there, and what are they doing? *Annu Rev Genet* **44**, 167-188.

Tomizawa, J., Itoh, T., Selzer, G., and Som, T. (1981). Inhibition of ColE1 RNA primer formation by a plasmid-specified small RNA. *Proc Natl Acad Sci U S A* **78**, 1421-1425.

Tuller, T., Kupiec, M., and Ruppin, E. (2007). Determinants of protein abundance and translation efficiency in S. cerevisiae. *PLoS Comput Biol* **3**, e248.

Tuller, T., Waldman, Y.Y., Kupiec, M., and Ruppin, E. (2010). Translation efficiency is determined by both codon bias and folding energy. *Proc Natl Acad Sci U S A* **107**, 3645-3650.

Urbanowski, M.L., Stauffer, L.T., and Stauffer, G.V. (2000). The gcvB gene encodes a small untranslated RNA involved in expression of the dipeptide and oligopeptide transport systems in Escherichia coli. *Mol Microbiol* **37**, 856-868.

Varenne, S., Buc, J., Lloubes, R., and Lazdunski, C. (1984). Translation is a non-uniform process. Effect of tRNA availability on the rate of elongation of nascent polypeptide chains. *J Mol Biol* **180**, 549-576.

Vellanoweth, R.L., and Rabinowitz, J.C. (1992). The influence of ribosome-binding-site elements on translational efficiency in Bacillus subtilis and Escherichia coli in vivo. *Mol Microbiol* **6**, 1105-1114.

Vimberg, V., Tats, A., Remm, M., and Tenson, T. (2007). Translation initiation region sequence preferences in Escherichia coli. *BMC Mol Biol* **8**, 100.

Vogel, C., Abreu Rde, S., Ko, D., Le, S.Y., Shapiro, B.A., Burns, S.C., Sandhu, D., Boutz, D.R., Marcotte, E.M., and Penalva, L.O. (2010). Sequence signatures and mRNA concentration can explain two-thirds of protein abundance variation in a human cell line. *Mol Syst Biol* **6**, 400.

Vogel, C., and Marcotte, E.M. (2012). Insights into the regulation of protein abundance from proteomic and transcriptomic analyses. *Nat Rev Genet* **13**, 227-232.

Wagner, A. (2008). Robustness and evolvability: a paradox resolved. *Proc Biol Sci* **275**, 91-100.

Wagner, E.G., Altuvia, S., and Romby, P. (2002). Antisense RNAs in bacteria and their genetic elements. *Adv Genet* **46**, 361-398.

Wang, H.H., Isaacs, F.J., Carr, P.A., Sun, Z.Z., Xu, G., Forest, C.R., and Church, G.M. (2009). Programming cells by multiplex genome engineering and accelerated evolution. *Nature* **460**, 894-898.

Wassarman, K.M. (2002). Small RNAs in bacteria: diverse regulators of gene expression in response to environmental changes. *Cell* **109**, 141-144.

Waters, L.S., and Storz, G. (2009). Regulatory RNAs in bacteria. *Cell* **136**, 615-628.

Welch, M., Govindarajan, S., Ness, J.E., Villalobos, A., Gurney, A., Minshull, J., and Gustafsson, C. (2009a). Design parameters to control synthetic gene expression in Escherichia coli. *PLoS One* **4**, e7002.

Welch, M., Villalobos, A., Gustafsson, C., and Minshull, J. (2009b). You're one in a googol: optimizing genes for protein expression. *J R Soc Interface* **6 Suppl 4**, S467-476.

Welch, M., Villalobos, A., Gustafsson, C., and Minshull, J. (2011). Designing genes for successful protein expression. *Methods Enzymol* **498**, 43-66.

Widmaier, D.M., Tullman-Ercek, D., Mirsky, E.A., Hill, R., Govindarajan, S., Minshull, J., and Voigt, C.A. (2009). Engineering the Salmonella type III secretion system to export spider silk monomers. *Mol Syst Biol* **5**, 309.

Wilkinson, B., and Micklefield, J. (2007). Mining and engineering natural-product biosynthetic pathways. *Nat Chem Biol* **3**, 379-386.

Win, M.N., and Smolke, C.D. (2008). Higher-order cellular information processing with synthetic RNA devices. *Science* **322**, 456-460.

Wold, S., Sjostrom, M., and Eriksson, L. (2001). PLS-regression: a basic tool of chemometrics. *Chemometr Intell Lab* **58**, 109-130.

Xie, Z., Wroblewska, L., Prochazka, L., Weiss, R., and Benenson, Y. (2011). Multi-input RNAi-based logic circuit for identification of specific cancer cells. *Science* **333**, 1307-1311.

Yager, T.D., and von Hippel, P.H. (1991). A thermodynamic analysis of RNA transcript elongation and termination in Escherichia coli. *Biochemistry* **30**, 1097-1118.

Yamao, F., Andachi, Y., Muto, A., Ikemura, T., and Osawa, S. (1991). Levels of tRNAs in bacterial cells as affected by amino acid usage in proteins. *Nucleic Acids Res* **19**, 6119-6122.

Yarchuk, O., Jacques, N., Guillerez, J., and Dreyfus, M. (1992). Interdependence of translation, transcription and mRNA degradation in the lacZ gene. *J Mol Biol* **226**, 581-596.

Yusupova, G.Z., Yusupov, M.M., Cate, J.H., and Noller, H.F. (2001). The path of messenger RNA through the ribosome. *Cell* **106**, 233-241.

Zhan, J., Ding, B., Ma, R., Ma, X., Su, X., Zhao, Y., Liu, Z., Wu, J., and Liu, H. (2010). Develop reusable and combinable designs for transcriptional logic gates. *Mol Syst Biol* **6**, 388.

Zhang, F., Cong, L., Lodato, S., Kosuri, S., Church, G.M., and Arlotta, P. (2011). Efficient construction of sequence-specific TAL effectors for modulating mammalian transcription. *Nat Biotechnol* **29**, 149-153.

Zhao, H., and Chen, W. (2008). Chemical biotechnology: microbial solutions to global change. Editorial overview. *Curr Opin Biotechnol* **19**, 541-543.

Appendix I

D-Tailor Tutorial

D-Tailor tutorial

Joao C Guimaraes, Miguel Rocha, Adam P Arkin and Guillaume Cambray

02/28/13

Index

1. Installing D-Tailor

1.1. Prerequisites

D-Tailor is implemented in Python. Python is an interpreted and interactive object-oriented programming language that is available for several platforms including Unix, Mac OSX and Microsoft Windows. Before starting to use D-Tailor, you need to install Python. More information can be found at http://www.python.org.

D-Tailor uses a few command line utilities such as *cat or awk* that are commonly available for Unix or Unix-derived operating systems. When using Microsoft Windows it may be necessary to run D-Tailor in a Unix-emulation environment such as Cygwin (http://www.cygwin.com/).

To have access to certain functionalities in D-Tailor, you will need to install third-party software to predict RNA structure (UNAFold v3.6 and RNAplfold v1.6) and transcription terminators (TransTermHP v2.08). The sources for these tools are located in the folder "3rdParty" and, after installation, the compiled binaries must be copied to each corresponding folder. For installation instructions, please refer to the respective websites:

- UNAFold — http://mfold.rna.albany.edu/?q=DINAMelt/software
- RNAplfold (Vienna RNA package) — http://www.tbi.univie.ac.at/~ivo/RNA/
- TransTermHP — http://transterm.cbcb.umd.edu/

All these tools are optional and hence only necessary if the user wants to use above-mentioned functionalities, namely predict RNA structure or transcription terminators.

1.2. Installation

D-Tailor is a Python project ready to be used. To start using D-Tailor, simply download it from http://sourceforge.net/projects/dtailor and copy the files to the destination folder.

1.3. License

D-Tailor is licensed under the BSD 2-Clause License.

2. D-Tailor Essentials

2.1. Scope and functionalities

D-Tailor is an extendable framework that automates the analysis and design of DNA sequences. To this end, it implements two distinct modules: Sequence Analyzer and Sequence Designer. In the Analyzer module, a predefined set of features of interest is inferred from plain DNA sequences. Conversely, the Designer module evolves DNA sequences to meet several properties of interest. In this mode, D-Tailor starts from an input seed sequence to iteratively generate sequence derivatives matching user-defined features scores under specified mutational constraints (e.g. positions available for mutation or only perform synonymous mutations). In addition to biasing the mutation process, users can also parameterize the strength of selection, which affects the overall diversity of the designed sequences, and further enforce validation tests to prevent final sequences from comprising undesired elements (e.g. restriction sites, unexpected promoters, terminators or internal ribosome binding sites).

In summary, D-Tailor provides an integrated framework for the seamless extraction of multiple features of interest from plain DNA sequences. This analysis pipeline is integrated in a Monte-Carlo architecture that is used to evolve input sequences under user-defined constraints toward a set of target combinations of features scores, thereby enabling multi-objective sequence design. D-Tailor is based on an extendable architecture to allow the independent development of new sequence features evaluators that can be easily plugged-in to the software (Figure 2.1).

2.2. Project Structure

D-Tailor uses object-oriented design and its core entities are:

- *Feature* —Abstract class encapsulating all relevant attributes and methods to describe a particular sequence feature or property;
- *Solution* — Concrete class containing all information concerning a particular sequence.

A *Solution* can have one or more *Feature* objects. *Solution* is a concrete class that can readily be used to store a DNA/RNA sequence and perform subsequent extraction of the features of interest. In contrast, *Feature* is an abstract class that should be extended to implement a concrete feature. D-Tailor comes packaged with many ready-to-use concrete features that extend the abstract class *Feature*. We will

use a detailed implementation of two of these features to exemplify how users can easily implement their own feature of interest (below).

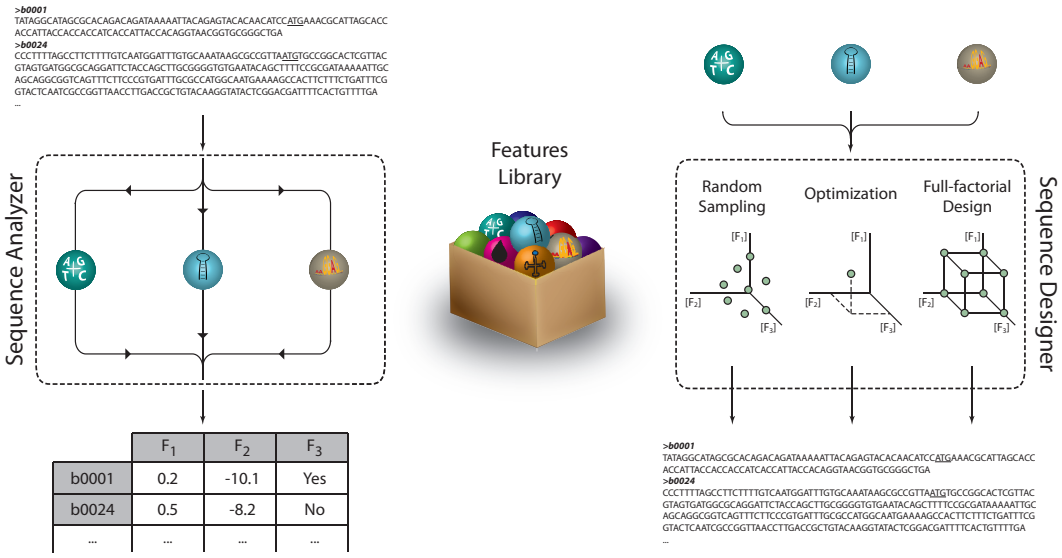


Figure 2.1 D-Tailor framework

D-Tailor provides automated analysis and design of DNA sequences. Because it is based on a modular architecture, it enables the independent development of sequence features evaluators (Features library) that can then be plugged-in to the software. The left panel depicts an example of the retrieval process of three features from DNA sequences. The right panel outlines the design mode of D-Tailor, wherein designed sequences are generated based on multiple features of interest and design schemes. The Features library (depicted in the middle) is used by both the Analyzer and Designer modules.

Figure 2.2 depicts a unified model language (UML) class diagram that captures the dependencies between classes implemented in D-Tailor.

The two main executable classes of D-Tailor are called *SequenceAnalyzer* and *SequenceDesigner*. These classes implement the basic functionality of the engine responsible for the analysis and design of sequences, respectively. These are abstract classes and need to be extended to implement user-defined analyses and designs (e.g. which features to compute or what region of the sequence can be mutated). The design mode requires the instantiation of an additional class that defines the design goal. This class must be an extension of the abstract class *Design*. Several design methodologies are already implemented in D-Tailor (see section 6).

To provide a convenient working environment and further enable parallel computation, results need to be permanently stored and easily accessed as sequences are generated and evaluated. D-Tailor contains an abstract class called *DBAbstract* that encapsulates a database management system interface where all the information regarding sequence information and features are stored. We have extended this class

to implement a storage environment based on SQLite (http://www.sqlite.org/). *DBSQLite* uses the built-in Python library *sqlite3* to implement a file-based SQL database engine that is used to store information resiliently and in a structured way without the need to install additional software for database management. Furthermore, it provides a common repository when multiple instances of the *SequenceDesigner* are run in parallel. Other database solutions can be implemented by extending *DBAbstract* to provide a storage environment compatible with the user preferences (e.g. SQLServer, MySQL, etc) without impacting the basic functionalities of D-Tailor.

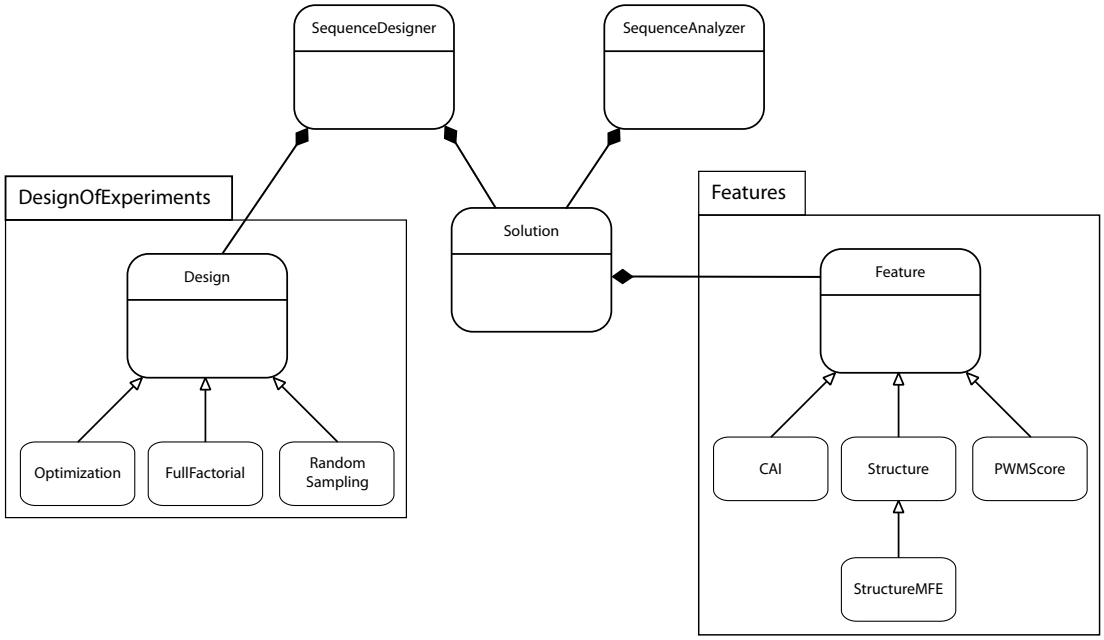


Figure 2.2 UML class diagram of D-Tailor

This diagram depicts the main classes implemented in D-Tailor. The two main programs *SequenceAnalyzer* and *SequenceDesigner* contain one or more instances of the class *Solution*, which contains a list of one or more instances of the class *Feature*. The *SequenceDesigner* also requires an instantiation of the class *Design,* which provides basic information about the target(s) one is trying to design for. The diagram also shows some examples of classes extending the abstract classes *Design* and *Feature*.

The project itself is organized in a series of folders and packages:

- Root directory: contains several core classes of the software, namely *SequenceAnalyzer*, *SequenceDesigner* and *Solution*. It also contains two auxiliary modules (*Data* and *Functions*) that contain a collection of relevant data structures and functions, respectively. *Data* is intended to contain all data variables/structures shared by the many classes and can be seen as a repository for global variables. For example, it contains a dictionary that can

be used to map codons to amino acids and vice-versa. The module *Functions* provides a repository for common functions that is shared by different classes.

- Packages:
  - *DBOperation* — contains the abstract and concrete classes implementing the storage management system;
  - *DesignOfExperiments* — includes all the classes defining the design methods that can be used by the *SequenceDesigner*;
  - *Features* — collection of the several features evaluators that are already implemented in D-Tailor;
  - *Running examples* — contains a few illustrative examples of how to use the two different modes of operation of D-Tailor (analysis and design);
  - *Utils* — a set of auxiliary tools (e.g. provide database statistics such as number of sequences generated or number of targets found, below).
- Folders:
  - *3rdParty* — folder containing 3rd party software that may be required to run certain feature evaluators (e.g. UNAFold to compute RNA structures);
  - *testFiles* — a collection of test files that are used by the running examples (e.g. *E. coli* genome);
  - *tmp* — a folder where temporary outputs generated by the sequence evaluators are saved (e.g. structure files produced by UNAFold);
  - *db* — where the databases generated in the design mode can be stored;
  - *class_diagram* — contains an UML class diagram of the D-Tailor project.

3. *Feature* class: handling features

The class *Feature* encapsulates the concept of a sequence feature (i.e. a property calculated from the raw sequence). It is intended to store all relevant information about the particular feature of a sequence, and also contains all the methods necessary to calculate the feature scores. In D-Tailor, *Feature* is an abstract class that must be extended by classes implementing features of interest. Figure 3.1 shows an example of a class that extends *Feature* for the calculation of the codon adaptation index (CAI). Basically, this CAI class only needs to implement a constructor, which has to call the super class constructor from *Feature* and further define specific attributes, and the method *set_scores* that is responsible for the computation of the feature scores. Importantly, the score of a given feature needs to be stored in a dictionary called *scores* using the appropriate key, which must be a string resulting from concatenating the given label and the name of the feature class (Figure 3.1).

```
class CAI(Feature):

    def __init__(self, caiObject = None, solution = None, label="",
                       args = { 'cai_range' : (0,59), 'mutable_region' : None,
                                'cds_region' : None , 'keep_aa' : True }):

        if caiObject == None: #create new instance
            #General properties of feature
            Feature.__init__(self, solution=solution, label=label)
            #Specifics of this Feature
            self.cai_range  = args['cai_range']
            self.sequence   = solution.sequence[self.cai_range[0]:(self.cai_range[1]+1)]
            self.set_scores()
            self.set_level()
        else: #copy instance
            …

    def set_scores(self, scoring_function=Functions.analyze_cai):
        self.scores[self.label+"CAI"] = scoring_function(self.sequence)
```

Figure 3.1 Definition of a class implementing a feature (CAI)

The constructor of this class receives three input parameters: a *Solution* (solution), a *string* (label) and a dictionary *args* with all the parameters necessary to configure the feature. In this case, to calculate CAI score, we only need the region of the sequence where we want to compute it. This parameter comes in the dictionary *args* and is accessed via the key *cai_range*. Once the required parameters are defined, we only need to implement the necessary steps to compute the feature score(s). In the class CAI, we have a private variable called *sequence* that will contain the exact sequence that needs evaluation. To perform the calculation, we implemented the method *analyze_cai* that computes the geometric mean of the weight associated with

each codon within *sequence* (Figure 3.2). To enhance software reusability, we decided to implement functions like this in the *Functions* module.

```
def analyze_cai(seq):
    seq = seq.lower();
    score = 0
    len_sq = 0
    for i in range(0,len(seq),3):
        if cai_table.has_key(seq[i:i+3]):
            score += log(cai_table[seq[i:i+3]])
            len_sq += 1
    score /= len_sq
    return exp(score)
```

Figure 3.2 Calculation of CAI score

Given the simplicity of CAI calculation, the *analyze_cai* could be directly implemented in python. However, many complex features require sophisticated algorithms that are already available in third party software. D-Tailor can also be used to provide a streamlined way to call such software. For example, we implemented a feature that uses the external command line tool UNAfold to compute the RNA secondary structure of a given sequence (Figure 3.3).

```
class Structure(Feature):

    def __init__(self, structureObject = None, solution = None, label="",
                        args = { 'structure_range' : (0,59), 'mutable_region' : None,
                                 'cds_region' : None,'keep_aa' : True }):

        if structureObject == None: #create new instance
            #General properties of feature
            Feature.__init__(self, solution=solution, label=label)
            #Specifics of this Feature
            self.structurefile      = solution.solid + label
            self.structure_range    = args['structure_range']
            self.sequence           =
                   solution.sequence[self.structure_range[0]:(self.structure_range[1]+1)]
            self.set_scores()
            self.set_level()
        else: #copy instance
            …

    def set_scores(self, scoring_function=Functions.analyze_structure):
        scoring_function(self.sequence, self.structurefile)
```

Figure 3.3 Definition of the class Structure

Similarly to the CAI feature, this class only implements the constructor and the *set_scores* method. Here, the parameter specifying the region of the sequence where the structure should be predicted is given by *structure_range*. Then, the function *analyze_structure* is responsible for calling the RNA structure prediction tool and process its output. In this case, the function saves the structure predicted by UNAfold with a predefined name (given by the variable *structurefile* in the main class Structure) and moves it to 'tmp/structures/' (Figure 3.4).

```
def analyze_structure(seq,filename,ensemble=False):

    chdir(project_dir)
    system("echo '" + str(seq) + "' > " + filename + ".seq")
    fnull = open(devnull, 'w')   # omit output generated by UNAFOLD
    if ensemble:
        call("./3rdParty/unafold/UNAFold.pl -n RNA " + filename + ".seq", shell = True, stdout =
fnull, stderr = fnull)
    else:
        call("./3rdParty/unafold/hybrid-ss-min -n RNA " + filename + ".seq", shell = True, stdout
= fnull, stderr = fnull)
    system("mv %s*.ct tmp/structures/" % filename)
    # remove tmp files
    system("rm %s*" % filename)
    fnull.close()

    return 1
```

Figure 3.4 Prediction of RNA structure using external software

The class Structure does not compute any specific score *per se*. This design pattern is useful whenever different feature scores can be derived from the same object, as it avoids re-instantiating the parent object many times that way saving computation time. For example, different scores, such as the minimum free energy or the bases that are paired, can be inferred from the same RNA secondary structure. In this case, sub-classes implementing these score calculations can extend a parent class according to a hierarchical class inheritance pattern. Figure 3.5 shows the class StructureMFE, which can be used to compute the minimum folding energy (MFE) for a structure predicted using the class Structure. Since this class computes a feature score, it is required to update the dictionary *scores* with the calculated score. In this particular example, the score is computed by calling another tool of the UNAFold package that calculates the MFE from a structure file (created previously by the Structure object).

In summary, a class extending the abstract class *Feature* will have the following attributes:

- *label* — a user defined label for the features;
- *solution* — object of class Solution where feature should be calculated;
- *subfeatures* — a dictionary with all sub-features associated with this feature (e.g. the two alternative structure features mentioned above);
- *scores* — a dictionary containing the score for the feature and its sub-features.

D-Tailor comes with several features implemented out-of-the-box. Most of them are directly related to sequence features impacting gene expression. As documented above, the software can easily be extended to implement any other feature of interest. A list of the features currently implemented in D-Tailor is depicted in the table below. Users are encouraged to contact the authors for help in the

implementation of new features or if they want to contribute new features to future releases of the package.

```
class StructureMFE(Structure):
    """
    Manipulate the structure MFE
    """
    def __init__(self, structureObject, label = "", regionOfInterest= None):
        Structure.__init__(self,structureObject)
        self.label = self.label + label
        self.set_scores()
        self.set_level()

    def set_scores(self, scoring_function=Functions.analyze_structure_mfe):
        self.scores.update(Functions.appendLabelToDict(scoring_function(self.structurefile),
self.label))


Functions.py:

def analyze_structure_mfe(filename,region = None):
    data = {}
    chdir(project_dir)

    if region == None:
        if path.exists(project_dir+"/tmp/structures/"+filename+".ct"):

            output = check_output(["./3rdParty/unafold/ct-energy" , "tmp/structures/" + filename +
                                   ".ct"]).rstrip()
            mfe_list = [float(a) for a in output.split('\n')]

            data['StructureMFE'] = mfe_list[0]
        else:
            data['StructureMFE'] = 0

    return data
```

Figure 3.5 Definition of the class StructureMFE extending Structure

Feature class	Description
CAI	Scores gene sequence codon usage as compared to that of highly expressed genes. Computes a score between 0 and 1, where the higher the score, the closer is the overall codon usage to the reference set. This feature only needs to be parameterized with *cai_range*, a pair of integers indicating the start and end nucleotide of the sub-sequence where the CAI should be calculated.
Structure	This feature uses UNAFold to predict the RNA secondary structure(s). Specifically, it uses UNAfold and redirects the generated structure-related files to the folder '*tmp/structures/*'. These can then be accessed by inheriting sub-classes that compute specific features scores (see below). The constructor of Structure only needs that the dictionary *args* specifies *structure_range*, a pair of integers indicating the start and end nucleotide of the sequence where the structure should be predicted.
StructureMFE	Extends the class Structure to retrieve the MFE structure score, as defined by the Gibbs free energy (ΔG). This feature is instantiated using an object of class *Structure*, which knows the path to the predicted MFE structure file. It essentially defines a UNAfold structure file parser that extracts the ΔG and updates the *scores* dictionary.
StructureSingleStranded	Extends the class Structure to compute a list, and count the total

	number, of the base positions that are single stranded (i.e. free) in the MFE structure.
StructureDoubleStranded	Extends the class Structure to compute a list, and count the total number, of the base positions that are double stranded (i.e. paired) in the MFE structure.
StructureEnsemble	This feature uses UNAFold to compute an ensemble of RNA structures instead of just the MFE. The result is list of structures that are saved to 'tmp/structures'. As in Structure, this class also needs the parameter *structure_range*. Note: This class does not compute any feature score.
StructureEnsembleAccessibility	Extends the class StructureEnsemble to calculate the average accessibility of each nucleotide (i.e. the probability of a nucleotide being free across all the putative structures formed in the ensemble) and the overall average.
StructureProb	This class uses the software *RNAplfold* from the Vienna RNA package to calculate the average probability of a base being unpaired in long sequences. This tool is particularly useful to probe a wide sequence range. The mandatory parameter *structure_range* indicates the start and end nucleotide positions of the subsequence to consider. Optionally, the parameters *window* and *acc_region* can be defined to configure the window size (by default 50) and a particular sub-region of interest where average accessibility should be calculated, respectively. The resulting *scores* contain a dictionary with the unpaired probability for each base (*XStructureProbList*) and the average probability across all bases (*XStructureProb*), where *X* is the label given to the feature
HydropathyIndex	This class calculates the average hydropathy index of a peptide based on the properties of its constitutive amino acids. Larger scores indicate more hydrophobic properties. The peptide sequence is translated from the DNA coding subsequence specified by the parameter *hi_range* — a pair of integers indicating the start and end of the subsequence.
NucleotideContent	This feature calculates the nucleotide content of a particular sequence (% of A, C, G, T, AT, GC)
RNADuplex	This class is intended to predict the hybridization of any two RNA molecules. As such, it receives two input sequences and uses UNAfold to predict the duplex formed. This structure is then saved to 'tmp/structures'.
RNADuplexRibosome	Extends the class RNADuplex to implement the interaction between an RNA molecule and the 16S rRNA.
RNADuplexMFE	Extends the class RNADuplex to calculate the MFE of the duplex.
PWMScore	This class implements the search for a subsequence with highest score using a given sequence motif (specified by a position weight matrix (PWM)). When initializing this class, two parameters have to be included in the *args* dictionary: (1) the region of the sequence to find the motif (*pwm_range*, a pair of integers), and (2) a PWM used to score the sequence (*pwm*). A PWM is a dictionary where the keys (1-letter conventional symbols for DNA, RNA or amino acids) are associated with a list of weights (one per position). D-Tailor comes with pre-configured PWMs for *E. coli*, namely for SD and promoter regions (see module *Data*).

4. *Solution* class: handling sequences

The class *Solution* is the realization of a particular sequence along with all the features of interest that were computed from it. *Solution* has the following basic attributes:

- *solid* — unique solution identifier;
- *sequence* — the full sequence to be analyzed, subsequences being specified in the specific features;
- *features* — a dictionary filled with pairs (Feature's label, Feature object);
- *scores* — a dictionary that aggregates the scores of all the features within this *Solution* (keys are the labels defined in features concatenated with features classes names);

Some of these attributes are instantiated when an object of type *Solution* is created, namely *solid* and *sequence*. Following the creation of a *Solution*, objects of type *Feature* can be added using the generic method *add_feature*. This method automatically updates the *features* and *scores* dictionaries. This way *Solution* objects can be easily created and further populated with an arbitrary set of features of interest (Figure 4.1).

```
>>> from Solution import Solution
>>> from Features.CAI import CAI

## Instantiate an object of type 'Solution'
>>> solution = Solution(sol_id = 'b0001', sequence = 'TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAA
CATCCATGAAACGCATTAGCACCACCATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGA')

## Instantiate Feature objects of interest
# Feature to calculates the codon adaptation index
>>> cai_obj = CAI(solution=solution,label="cds",args= {'cai_range':(49,115)})
#Feature to predicts RNA Structure
>>> st1_obj = Structure(solution=solution ,label="utrCds",args= { 'structure_range' : (19,78) } )
# Two sub-features inheriting from the class Structure
>>> st_mfe = StructureMFE(st1_obj)
>>> st_ss  = StructureDoubleStranded(st1_obj)
>>> st1_obj.add_subfeature(st_mfe)
>>> st1_obj.add_subfeature(st_ss)

# Feature to calculate nucleotide content
>>>nuc_obj = NucleotideContent(solution=solution ,label="utr",args= { 'ntcontent_range':(0,50) } )

## Add features to solution
>>> solution.add_feature(cai_obj)
>>> solution.add_feature(st1_obj)
>>> solution.add_feature(nuc_obj)

## Retrieve feature score
>>>  solution.scores
{'cdsCAI': 0.6136121593930156,
 'utrCdsStructureDoubleStrandedList':[18, 19, 25, 26, 38, 39, 44, 45],
 'utrCdsStructureDoubleStranded': 8, 'utrCdsStructureMFE': -2.5,
 'utrNucleotideContentAT': 0.63,  'utrNucleotideContentG': 0.16,'utrNucleotideContentT' : 0.18,
 'utrNucleotideContentC': 0.22, 'utrNucleotideContentA' : 0.45, 'utrNucleotideContentGC': 0.37}
```

Figure 4.1 Definition of an object Solution with multiple features

5. Sequence Analyzer

The sequence analyzer mode of D-Tailor provides an integrated solution for the multidimensional interrogation of sequences. For that, the user needs to extend the abstract class *SequenceAnalyzer* and implement the required methods. This abstract class can read the sequences of interest from CSV and FASTA files (CSV files must contain the headers 'name' and 'sequence'). Then, the method *run()* simply loops through the loaded sequences and computes the features of interest (Figure 5.1).

```
def run(self):

    self.outputStart()

    for sequence in self.list_of_input_sequences:
        sol_id = sequence['name']
        seq = sequence['sequence']

        solution = Solution(sol_id = sol_id, sequence = seq)
        self.configureSolution(solution)

        self.output(solution)
```

Figure 5.1 Method *run()* in *SolutionAnalyzer*

When implementing a class extending *SequenceAnalyzer*, the user is required to implement the following methods:

- *configureSolution* — this method instantiates the features to compute on a given sequence (Solution). Its architecture is similar to what is shown in Figure 4.1;
- *outputStart* —called once at the beginning of the method run() to initialize the output (e.g. open a file and/or write an header);
- *output* — this operation is performed after the configuration step and can be used to perform operations on the retrieved features (e.g. print to the screen).

Figure 5.2 shows the class *TranslationFeaturesEcoli* (located in *RunningExamples/Analyzer*) that extends *SequenceAnalyzer* and calculates the three different features impacting translation efficiency in *E. coli*:

- CAI — a proxy for the translation elongation rate along the gene;
- Hybridization energy between Shine-Dalgarno (SD) region and 16S rRNA;
- RNA Structure around translation initiation region.

```
from SequenceAnalyzer import SequenceAnalyzer
from Features import CAI,Structure,RNADuplex
from Functions import validateCDS

class TranslationFeaturesEcoliAnalyzer(SequenceAnalyzer):

    '''
    Class to analyze CAI, SD strength and structure in E. coli
    '''

    def __init__(self, input_file, input_type):
        SequenceAnalyzer.__init__(self,input_file,input_type)

    def configureSolution(self, solution):
        solution.valid = validateCDS(solution.sequence[49:])

        if solution.valid:
            #CAI
            cai_obj = CAI.CAI(solution=solution,label="cds",args= {  'cai_range' :
                                                                     (49,len(solution.sequence)) }
)

            #Look for RBS
            dup_obj1 = RNADuplex.RNADuplexRibosome(solution1=solution, label="sd16s",
                                                   args = { 'rnaMolecule1region' : (25,48)
})
            dup_mfe = RNADuplex.RNADuplexMFE(dup_obj1)
            dup_obj1.add_subfeature(dup_mfe)

            #MFE [-30,30]
            st1_obj = Structure.Structure(solution=solution,label="utr",
                                                 args= { 'structure_range' : (49-30,49+30) }
)
            st_mfe = Structure.StructureMFE(st1_obj)
            st1_obj.add_subfeature(st_mfe)

            solution.add_feature(cai_obj)
            solution.add_feature(dup_obj1)
            solution.add_feature(st1_obj)

    def outputStart(self):
        print "gene_name,sd_hyb_energy,mfe_structure,cai"

    def output(self, solution):
        if solution.valid:
            print solution.solid,",",
                  solution.scores['sd16sRNADuplexMFE'],",",
                  solution.scores['utrStructureMFE'],",",
                  solution.scores['cdsCAI']

if __name__ == '__main__':
    seqAnalyzerTest = \

TranslationFeaturesEcoli("../../testFiles/genomes/partial_ecoli_genome.csv","CSV")
    seqAnalyzerTest.run()
```

Figure 5.2 Class *TranslationFeaturesEcoliAnalyzer* calculates multiple features for all E. coli genes

This class loads all *E. coli* coding regions along with the 49 nucleotides preceding them into the sequence analyzer module (this table can be found at *testFiles/genomes/ecoli_genome.csv*). The three features of interest are defined in the *configureSolution* method, which also checks if the provided coding sequences are valid (i.e. has start codon and no in-frame stop codons). The user can further define output options using the methods *outputStart* and *output*. In this example, we are simply printing the computed features to the screen from the *Solution* object *scores* dictionary (note that the key for each score is the label of the feature concatenated

with the feature class name, e.g. 'utrStructureMFE'). The program above will print to the screen a table-like output that is partially shown in Figure 5.3.

```
macbook:D-Tailor jcg$ PYTHONPATH=. python
RunningExamples/Analyzer/TranslationFeaturesEcoliAnalyzer.py
gene_name,sd_hyb_energy,mfe_structure,cai
b0001,-1.8,-1.925,0.613612159393
b0002,-7,-9.16,0.34043688741
b0003,-5.7,-13.2,0.341658034933
b0004,-3.2,-5.5,0.385891327353
b0005,-7.3,-6.76,0.377281853234
b0006,-6.1,-14.75,0.342733396212
b0007,-5.6,-8.4,0.319183029826
b0008,-2.7,-8.1,0.604195702312
b0009,-3.5,-7.5,0.396623675448
b0010,-6.2,-9.6,0.574062247682
b0011,-0.3,-4.5,0.286738246339
b0013,-2.3,-7.8,0.362374253526
b0014,-5.4,-7.08333333333,0.723381361599
b0015,-4.8,-6.45,0.525547136369
…
```

Figure 5.3 Partial output of *TranslationFeaturesEcoliAnalyzer*

This output can then be easily imported into statistical tools such as SciPy or R for posterior analysis. For instance, Figure 5.4 shows how the three different features are distributed and related with each other.

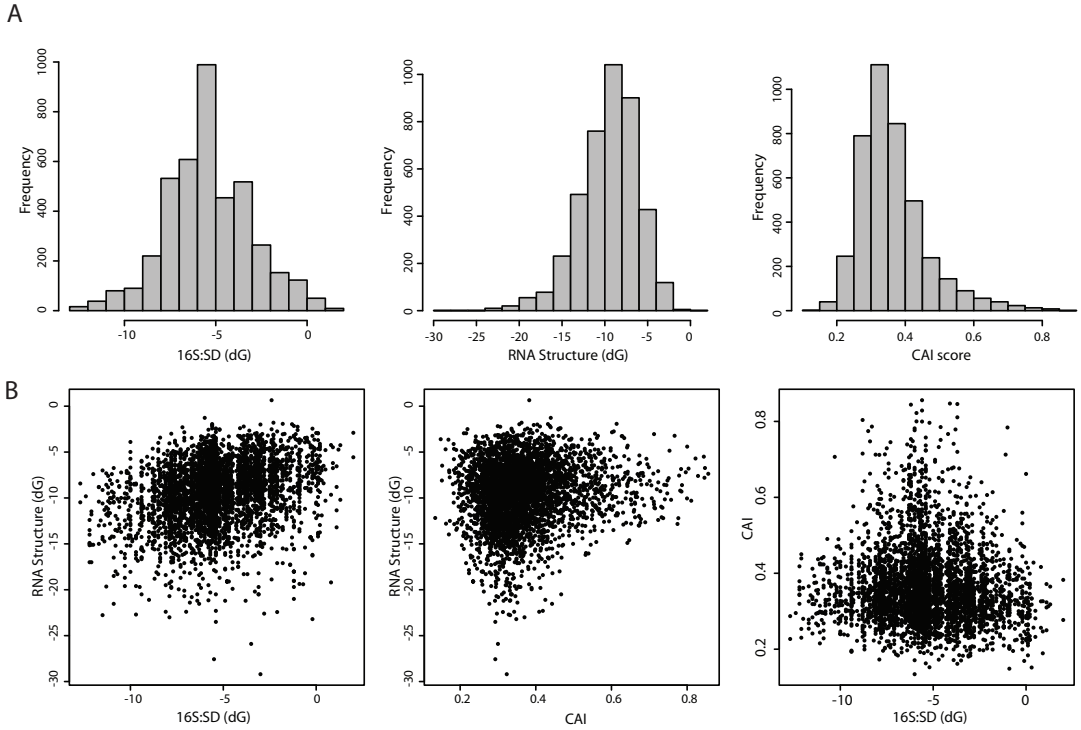


Figure 5.4 Distribution and relationship of translation features in *E. coli*

6. Sequence Designer

The most innovative functionality of D-Tailor is the ability to design sequences that meet user-defined goals. This section provides a detailed description of how to define a class extending *SequenceDesigner* to use this functionality. Briefly, the user needs to provide a seed sequence (from which the designed sequences will be derived), the features to be computed for each sequence, a design goal (a target combination of desired features scores, or a list of such targets) and a database file (where all the generated sequences and respective features are stored). Additionally, we describe multiple parameters by which users can constrain the way sequences are mutated and selected.

This chapter demonstrates how to configure a design module for the three features analyzed in the previous chapter to design new sequences. The code used in this section can be found in the class *TranslationFeaturesEcoliDesigner* located in the package *RunningExamples/Designer*.

6.1. Definition of features

Users first need to create a class extending *SequenceDesigner.* Similarly to the extension of the *SequenceAnalyzer* class (above), the concrete class has to implement the method *configureSolution* where all the features classes are instantiated and associated with a given *Solution* (Figure 6.1). *SequenceDesigner* requires the declaration of three additional parameters to guide the design process:

- *mutational_region* — a list of all the positions that have permission to be mutated;
- *cds_region* — a pair of integers defining the positions of the coding sequence, if any, to inform the program when trying to mutate a gene;
- *keep_aa* — a Boolean indicating whether only synonymous mutations should be performed within coding region.

Different features can be configured with different mutational regions. This permits to target mutations toward regions of the sequence that are *a priori* susceptible to affect the feature score more effectively and therefore speed up the search algorithm. For example, in the example below, we define the *mutable_region* for the 16S:SD hybridization energy feature to be only the region comprised [25,48] nucleotide, i.e. where the SD sequence is located. We do this because mutations in this region are more likely to change the score of this feature than if done anywhere

else in the sequence. We use the term 'targeted' mutagenesis to refer to this guided procedure.

```
    def configureSolution(self, solution):
        '''
        Solution configuration
        '''
        if solution.sequence == None:
            return 0

        ## Designer specific

        solution.mutable_region=range(0,len(solution.sequence)) # whole region
        solution.cds_region = (49,len(solution.sequence))
        solution.keep_aa = True

        ## Populate solution with desired features

        # CAI
        cai_obj = CAI.CAI(solution = solution,label="cds",
                              args = {  'cai_range' : (49,len(solution.sequence)),
                                        'mutable_region' : range(49,len(solution.sequence)) } )

        # Search SD
        dup_obj1 = RNADuplex.RNADuplexRibosome(solution1=solution, label="sd16s",
                                    args = { 'rnaMolecule1region' : (25,48),
                                             'mutable_region' : range(25,48) })
        dup_mfe = RNADuplex.RNADuplexMFE(dup_obj1)
        dup_obj1.add_subfeature(dup_mfe)

        # MFE [-30,30]
        st1_obj = Structure.Structure(solution=solution,label="utr",
                                    args = { 'structure_range' : (49-30,49+30)
                                             'mutable_region' : range(49-30,49+30)} )
        st_mfe = Structure.StructureMFE(st1_obj)
        st1_obj.add_subfeature(st_mfe)

        solution.add_feature(cai_obj)
        solution.add_feature(dup_obj1)
        solution.add_feature(st1_obj)
```

Figure 6.1 Definition of the method *configureSolution* in *TranslationFeaturesEcoliDesigner*

This code shows the definition of three sequence features to be computed for a *Solution.* In design mode, some constraints have to be defined to guide mutational process. These can be defined in the *Solution* object or at the *Feature* level (e.g. CAI). In this example, we are declaring that the entire solution region can be mutated, i.e. from position 0 to the length of the entire sequence. Additionally, we define that the gene starts at position 49 and ends at the end of sequence region (attribute *cds_region*), and that we only want to perform synonymous mutations (*keep_aa = True*). Since CAI is only affected by mutations within the coding region, we override the mutational constraints for this particular feature and indicate that to alter this feature we should perform mutations only within the gene sequence by initializing the parameter *mutable_region* in the instantiation of the class CAI.

In some cases, a good knowledge of the relationship between sequence and feature scores might allow to devise smart operators that 'rationally' guide the mutation process and increase the likelihood of producing a new sequence with a score closer to the desired target. We refer to this procedure as 'oriented' mutagenesis. For example, if meeting the target requires CAI to increase, a smart operator can readily choose an alternative codon with a higher score by looking up the CAI table. Similar approach can also be implemented for less deterministic features. For instance, we defined an oriented mutational scheme for the feature structure, wherein only paired bases in

the template structure are mutated when weaker structure is desired (and conversely for stronger structure) (Figure 6.2). Oriented mutational strategies provide some improvements over random mutation operators (below), and therefore should be implemented whenever possible.

The default mutational operator defined in the abstract class *Feature* implements the 'targeted' mutational operator with equiprobable mutation at all predefined mutable positions. When developing a new feature, users can override this operator with an oriented one by implementing the method *mutate* in the respective feature (Figure 6.2). Specific instruction regarding the direction the target score should be defined in the method *defineTarget* and stored in the class variable *targetInstructions* (note that abstract class *Feature* implements a minimal version of this method, where the direction is set to '+' if increasing the feature score is needed, or '-' otherwise) (Figure 6.2).

```
Feature.py:

    def defineTarget(self,desiredSolution):
        '''
        Function that determines if a target wasn't hit and, if not, updates target instructions
        '''
        if desiredSolution == None:
            return True

        #check if there is a target
        if not desiredSolution.has_key(self.label+self.__class__.__name__+"Level"):
            return False
        else:
            target_level = desiredSolution[self.label+self.__class__.__name__+"Level"]

            if target_level == 0:
                return False

            if  target_level != self.level:
                level_info =

self.solution.designMethod.thresholds[self.label+self.__class__.__name__][target_level]

                if isinstance(level_info, tuple): #Then it's a numeric range
                    if  level_info[0]-self.scores[self.label+self.__class__.__name__] > 0:
                        self.targetInstructions['direction'] = '+' #increase
                    elif level_info[0]-self.scores[self.label+self.__class__.__name__] < 0:
                        self.targetInstructions['direction'] = '-' #decrease
                else:
                    self.targetInstructions['direction'] = 'NA' #not applicable

                return True

            return False


Structure.py:

    #Overriding the mutation method to implement oriented mutation
    def mutate(self, operator=Functions.SimpleStructureOperator):
        if not self.targetInstructions:
            return None
        ss_bases = None if not self.scores.has_key(self.label+'StructureSingleStrandedBasesList')
else self.scores[self.label+'StructureSingleStrandedBasesList']
        ds_bases = None if not self.scores.has_key(self.label+'StructureDoubleStrandedBasesList')
else self.scores[self.label+'StructureDoubleStrandedBasesList']
        new_seq = operator(self.solution.sequence, self.structurefile,  self.structure_range,
```

```
self.mutable_region, self.cds_region, self.targetInstructions['direction'], ss_bases=ss_bases,
ds_bases=ds_bases)
        if not new_seq:
            return None
        return Solution.Solution(sol_id=str(uuid4().int), sequence=new_seq, cds_region =
self.cds_region, mutable_region = self.mutable_region, parent=self.solution,
design=self.solution.designMethod)


Functions.py:

def SimpleStructureOperator(sequence, structurefile, structure_range, mutable_region, cds_region,
direction, keep_aa = True, ss_bases=None, ds_bases=None):

    if not mutable_region: #it's not possible to mutate
        return None

    # get single stranded bases
    if ss_bases == None:
        ss_bases = structureAnalysis(structurefile, "ss")
    # get double stranded bases
    if ds_bases == None:
        ds_bases = structureAnalysis(structurefile, "ds")

    # for structure, increasing structure score (MFE) (+) means that we want to produce weaker
structures, so we will mutate double stranded bases
    if direction == '+':
        #get double stranded bases
        baseToMutate = [(b+structure_range[0]-1) for b in ds_bases \
                                                  if (b+structure_range[0]-1) in mutable_region]
    # conversely to increase structure we mutate single stranded bases
    elif direction == '-':
        #get single stranded bases
        baseToMutate = [(b+structure_range[0]-1) for b in ss_bases \
                                                  if (b+structure_range[0]-1) in mutable_region]
    else:
        sys.stderr.write("Direction Unknown")

    mutated = False
    iteration = 0

    #try to mutate up to 100 different times
    while not mutated and iteration <= 100:
        #select a position to mutate at random
        index_to_mutate = baseToMutate.pop(randint(0,len(baseToMutate)-1)) if len(baseToMutate) !=
0 else mutable_region.pop(randint(0,len(mutable_region)-1))

        #mutate base keeping amino acids (omitted)
        if keep_aa == True and index_to_mutate >= cds_region[0] and index_to_mutate <=
cds_region[1]:
            …
        #mutate without keeping amino acids
        else:
            mutated = True
            new_seq = list(sequence)
            if direction == '+':
                comp = complementary(sequence[index_to_mutate])
            else:
                comp = randomMutation(sequence[index_to_mutate])
            new_seq[index_to_mutate] = comp
            #print sequence
            #print "".join(new_seq)

        iteration+=1

    return "".join(new_seq)
```

Figure 6.2 Definition of the method *mutate* in *Structure*

The method *mutate* can be overridden to implement oriented mutations. For that, it may often be useful to have extra mutational information, such as the direction we want to alter the feature score (e.g. increase or decrease). These parameters are defined in the method *defineTarget*. A simplistic version of this method is already implemented in the abstract class *Feature*, which simply defines the

desired direction based on the current and target level. Finally, the code implementing oriented structure mutations is depicted in the figure. Here, we mutate either paired or unpaired bases if we want to decrease or increase structure strength, respectively.

6.2. Defining design goals

After the user defines the features of interest, it is necessary to define a design goal for D-Tailor. A design goal can be one or more target combinations of features scores. Alternatively, the design goal can simply define a number of sequences to be generated without any specific target combination (random sampling). In D-Tailor, a class used to define an experimental design needs to extend the abstract class *Design* and there is already four predefined methods:

- Optimization — only one specific target combination of features is desired. For example, to increase the expression of a given gene, we may want to design a sequence with high CAI, strong binding between SD and the 16S rRNA and weak mRNA structure around the initiation region.
- Full-Factorial — all possible combinations between the levels of the different features should be generated. This methodology is appropriate when one is not sure how features affect the observed phenotype and, hence, it is necessary to systematically vary these features to reliably infer new knowledge about the observed output.
- Custom Design — this is a more flexible design where the user can indicate one or more target combination(s) of features scores that he/she might want to design for.
- Random Sampling — this method does not enforce any particular combination of features *a priori*. It can be used to generate a predetermined number of new sequences and observe how they scatter across the feature space.

Design methods are based on the concept of feature levels, which are obtained by discretization of the feature scores (based on values intervals) and are necessary to yield finite designs. The user freely defines the number of levels for each feature. Here, the more levels are defined, the higher the resolution (and the smaller the predicted functional difference between features).

A set of features and their respective levels need to be inputted to a newly instantiated sub-class of *Design*. This is given in the form of a dictionary, where for each feature it will be necessary to define a type (REAL, INTEGER or CHAR) and a list of levels containing the respective lower and upper bounds (Figure 6.3). The user can freely decide how to divide the levels for each feature, but most likely he/she will

decide based on the analysis of a natural genome. For example, we can use three translational features analyzed in the previous chapter to design new sequences containing specific design goals (i.e. combination of features scores). First, we need to decide in how many levels we want to split each of the features of interest. Here, we chose to divide them into 5 categorical levels (very low, low, medium, high and very high) and used the quintiles identified in the genomic analysis to define their boundaries (Table 1).

Table 1 Definition of feature levels

	Very low (0-20%)	**Low (20-40%)**	**Medium (40-60%)**	**High (60-80%)**	**Very high (80-100%)**
16S:SD	[-12.7, -7.3[	[-7.3, -5.8[	[-5.8, -5.2[	[-5.2, -3.3[	[-3.3,2.0]
RNA structure	[-29.2, -12.2[	[-12.2, -9.95[	[-9.95, -8.4[	[-8.4, -6.73[	[-6.73, 0.65]
CAI	[0.13, 0.29[	[0.29, 0.33[	[0.33, 0.37[	[0.37, 0.42[	[0.42, 0.86]

Second, we need to define a design goal by instantiating one class of type *Design*. The design methods implemented in D-Tailor are located in the package *DesignOfExperiments*. A class implementing a design methodology has to extend the abstract class *Design* and does not need to implement any specific method but is required to initialize the following attributes:

- *listOfDesigns* – this is a vector of strings where each string is a particular combination of feature levels (e.g. '1.1.1' indicates a combination where all features scores are within the level 1 boundaries as defined in Table 1).
- *nDesigns* – the total number of different designs/combinations desired.

Figure 6.3 shows the definition of a full-factorial design by instantiating the class *FullFactorial*, which was parameterized with the three sequence features and the respective level thresholds (level identifiers should be numbers in ascending order). To perform a full-factorial design it will be necessary to generate all the combinations between the 5 levels for each of the 3 features (i.e. $5*5*5 = 5^3 = 125$ combinations).

```
>>> from DesignOfExperiments.Design import FullFactorial

#Design Methodology and thresholds
>>> design_param = {  "sd16sRNADuplexMFE": { 'type' : 'REAL' ,
                                            'thresholds' : { '1': (-12.7,-7.3), '2': (-7.3,-5.8),
                                                             '3': (-5.8,-5.2),  '4': (-5.2,-3.3),
                                                             '5': (-3.3, 2.0) } },
                      "utrStructureMFE"  : { 'type' : 'REAL' ,
                                            'thresholds' : { '1': (-29.2,-12.2), '2': (-12.2,-
9.95),
                                                             '3': (-9.95,-8.4),  '4': (-8.4,-
6.73),
                                                             '5': (-6.73,0.65) } },
                      "cdsCAI"           : { 'type' : 'REAL' ,
                                            'thresholds' : { '1': (0.13,0.29), '2': (0.29,0.33),
                                                             '3': (0.33,0.37), '4': (0.37,0.42),
                                                             '5': (0.42,0.86) } }
                  }

>>> design = FullFactorial(["sd16sRNADuplexMFE","utrStructureMFE","cdsCAI"],design_param)
>>> design.nDesigns
125
>>> design.listDesigns
['1.1.1', '1.1.3', '1.1.2', '1.1.5', '1.1.4', '1.3.1', '1.3.3', '1.3.2', '1.3.5', '1.3.4',
'1.2.1', '1.2.3', '1.2.2', '1.2.5', '1.2.4', '1.5.1', '1.5.3', '1.5.2', '1.5.5', '1.5.4', '1.4.1',
'1.4.3', '1.4.2', '1.4.5', '1.4.4', '3.1.1', '3.1.3', '3.1.2', '3.1.5', '3.1.4', '3.3.1', '3.3.3',
'3.3.2', '3.3.5', '3.3.4', '3.2.1', '3.2.3', '3.2.2', '3.2.5', '3.2.4', '3.5.1', '3.5.3', '3.5.2',
'3.5.5', '3.5.4', '3.4.1', '3.4.3', '3.4.2', '3.4.5', '3.4.4', '2.1.1', '2.1.3', '2.1.2', '2.1.5',
'2.1.4', '2.3.1', '2.3.3', '2.3.2', '2.3.5', '2.3.4', '2.2.1', '2.2.3', '2.2.2', '2.2.5', '2.2.4',
'2.5.1', '2.5.3', '2.5.2', '2.5.5', '2.5.4', '2.4.1', '2.4.3', '2.4.2', '2.4.5', '2.4.4', '5.1.1',
'5.1.3', '5.1.2', '5.1.5', '5.1.4', '5.3.1', '5.3.3', '5.3.2', '5.3.5', '5.3.4', '5.2.1', '5.2.3',
'5.2.2', '5.2.5', '5.2.4', '5.5.1', '5.5.3', '5.5.2', '5.5.5', '5.5.4', '5.4.1', '5.4.3', '5.4.2',
'5.4.5', '5.4.4', '4.1.1', '4.1.3', '4.1.2', '4.1.5', '4.1.4', '4.3.1', '4.3.3', '4.3.2', '4.3.5',
'4.3.4', '4.2.1', '4.2.3', '4.2.2', '4.2.5', '4.2.4', '4.5.1', '4.5.3', '4.5.2', '4.5.5', '4.5.4',
'4.4.1', '4.4.3', '4.4.2', '4.4.5', '4.4.4']
```

Figure 6.3 Defining a class of type Design (Full-Factorial)

6.3. Database of solutions

D-Tailor uses an SQLite database engine to store the generated solutions during the design process. This database contains three different tables:

- desired_solution — this is a dynamic table that is created on-the-fly during initialization of the database and contains all the user-defined combinations of features;
- generated_solution — this is also dynamic table where all the sequences generated are stored along with the feature scores and levels;
- worker — this stable stores the SequenceDesigner programs that ran on the database.

D-Tailor can be easily extended to other database engines. For that, it is only necessary to create a class that extends the abstract class *DBAbstract* and implements the required methods (Figure 6.4).

```
class DBAbstract(object):
    '''
    Abstract class for DNA Sequence Designer Database
    '''

    def DBInit(self):
        '''
        Initialize database
        returns: Nothing
        '''
        pass

    def DBGetSolution(self, solution_id):
        '''
        Get solution given solution_id
        returns: a dictionary with a solution with all attributes
        '''
        pass

    def DBGetDesiredSolution(self):
        '''
        Get a desired solution that wasn't found yet
        returns: a dictionary with a desired solution or None
        '''
        pass

    def DBChangeStatusDesiredSolution(self,desired_solution_id,status):
        '''
        Change the status of a desired solution
        '''
        pass


    def DBGetClosestSolution(self, desiredSolution):
        '''
        Get a solution that is closer to the desired solution
        returns: a dictionary with a solution with all attributes
        '''
        pass

    def DBCheckDesign(self, desired_solution_id):
        '''
        Get the status of a solution to design
        returns: a boolean with the result of status == 'DONE'
        '''
        pass

    def DBInsertSolution(self, solution, desired_solution_id):
        '''
        Insert solution into database
        returns: Nothing
        '''
        pass

    def DBCloseConnection(self):
        '''
        Closes connection to DB
        '''
        pass
```

Figure 6.4 Class *DBAbstract*

6.4. Running the designer

To start running the algorithm we need a *seed* sequence from which the designed sequences will be generated, a design method (from the class *Design*) indicating the design goals, a file path to where the database containing all the generated sequences

will be saved (Figure 6.5). In the example below, we used a *seed* sequence that contains a 5'UTR of 49 nucleotides including a SD region, followed by the gene sequence encoding human insulin (NCBI: NM_000207.2).

```
>>> from RunningExamples.Designer.TranslationFeaturesEcoliDesigner import
TranslationFeaturesEcoliDesigner
>>> from DesignOfExperiments.Design import FullFactorial

#Seed sequence from which mutants will be derived
>>>
seed='ttattaccggacaataatatttcaattcattaaagaggagaaaggtaccatggccctgtggatgcgcctcctgcccctgctggcgctgctgg
ccctctggggacctgacccagccgcagcctttgtgaaccaacacctgtgcggctcacacctggtggaagctctctacctagtgtgcggggaacgaggc
ttcttctacacacccaagacccgccgggaggcagaggacctgcaggtggggcaggtggagctgggcgggggccctggtgcaggcagcctgcagcccctt
ggccctggaggggtccctgcagaagcgtggcattgtggaacaatgctgtaccagcatctgctccctctaccagctggagaactactgcaactag'

#Design Methodology and thresholds
>>> design_param = {"sd16sRNADuplexMFE": { 'type' : 'REAL' ,
                                          'thresholds' : { '1': (-12.7,-7.3), '2': (-7.3,-5.8),
                                                           '3': (-5.8,-5.2),  '4': (-5.2,-3.3),
                                                           '5': (-3.3, 2.0) } },
                    "utrStructureMFE"  : { 'type' : 'REAL' ,
                                          'thresholds' : { '1':(-29.2,-12.2), '2': (-12.2,-9.95),
                                                           '3': -9.95,-8.4),  '4': (-8.4,-6.73),
                                                           '5': (-6.73,0.65) } },
                    "cdsCAI"           : { 'type' : 'REAL' ,
                                          'thresholds' : { '1': (0.13,0.29), '2': (0.29,0.33),
                                                           '3': (0.33,0.37), '4': (0.37,0.42),
                                                           '5': (0.42,0.86) } }
                   }

>>> design = FullFactorial(["sd16sRNADuplexMFE","utrStructureMFE","cdsCAI"],design_param)

>>> tirap_designer = TranslationFeaturesEcoliDesigner("tfec", seed, design,
"/Users/jcg/Documents/workspace/D-Tailor/testFiles/outputFiles/tfec_1", createDB=True)
>>> tirap_designer.run()
```

Figure 6.5 Running the *SequenceDesigner*

Please remember that the regions of the seed sequence that can be mutated were already defined in the class method *configureSolution* (Figure 6.1). Additionally, the user may also want to implement the method *validateSolution*, which is called every time a new sequence is generated. This validation step is fundamental to avoid undesired properties that the new mutated sequence may have (e.g. some restriction site that we may want to use for cloning). Our exemplary class *TranslationFeaturesEcoliDesigner* implements a series of validation tests that one may need when generating new sequences (Figure 6.6). Specifically, it checks if the new designed sequence does not include internal promoters, terminators and undesirable restriction enzymes sites (in this case BsaI sites).

```
def validateSolution(self, solution):
    '''
    Solution validation tests
    '''
    if solution.sequence == None or ('?' in solution.levels.values()):
        solution.valid = False
        return 0

    #check if solution is valid
    valid = True

    designed_region = solution.sequence

    #No internal Promoters
    (score, position, spacer) = Functions.look_for_promoters(designed_region)
    if score >= 15.3990166: #~0.95 percentile for Promoter PWM scores
        valid = False
        sys.stderr.write("SolutionValidator: High Promoter score\n")

    #No internal Terminator
    score = Functions.look_for_terminators(designed_region)
    if score >= 90: #90% confidence from transtermHP
        valid = False
        sys.stderr.write("SolutionValidator: High Terminator score\n")

    #No BsaI sites
    if 'ggtctc' in designed_region or 'gagacc' in designed_region:
        sys.stderr.write("SolutionValidator: Restriction enzyme found\n")
        valid = False

    solution.valid = valid

    return valid
```

Figure 6.6 Definition of the method *validateSolution*

When a new generated sequence does not pass the validation step, it will be discarded and not stored in the database. All the other generated sequences that pass the validation test will be stored in the database indicated by the user. The parameter *createDB* in the *SequenceDesigner* constructor should be set to 'True' when one wants to create an empty database (this will erase any existing database with the same name and create a new one). Otherwise, if the database is already created and we want to resume the designer algorithm or start multiple concurrent algorithms, we need to set this parameter to 'False'.

When the method *run()* is invoked in *SequenceDesigner*, the program will only stop when the design goal is achieved (e.g. in a full-factorial method the program will only exit when all target combinations are found). The program outputs, in real--time, the particular target combination the designer is looking for and statistics about how many sequences were already generated (Figure 6.7). When the Optimization design is selected, the program will additionally print the sequence that achieved the desired combination (Figure 6.8).

```
macbook:D-Tailor jcg$ PYTHONPATH=. python
RunningExamples/Designer/TranslationFeaturesEcoliDesigner.py fullfactorial
looking for combination:  2.3.5
SolutionValidator: Restriction enzyme found
No solution could be found...
looking for combination:  3.2.5
No solution could be found...
looking for combination:  4.4.3
No solution could be found...
time elapsed: 76.58 (s)    solutions generated: 385          rate (last min.): 5.03 sol/s
rate (overall): 5.03 sol/s
looking for combination:  3.5.3
…
Program finished...
```

Figure 6.7 Running *TranslationFeatureEcoliDesigner* (FullFactorial design)

```
macbook:D-Tailor jcg$ PYTHONPATH=. python
RunningExamples/Designer/TranslationFeaturesEcoliDesigner.py optimization 1.2.3
looking for combination:  1.2.3
Solution found... inserting into DB...

############################
# Optimized solution:
# ID:  461247999753940096228031914270368185O8
# Sequence:
ttattaccggacaataatatttcaattcattaaagaggagaaaggtaccatggcactttggatgcgcctcctgcccttactggcattactggcgctgt
ggggccctgacccggccgccgccttcgtgaatcaacatctgtgcggatcacacttggttgaggctctttacttagtgtgcggggaacgcggttttttc
tacacaccaaaaacgcgccgggaagcagaagacctgcaggttgggcaggtagaattaggtgggggccctggtgctggcagcctgcagcccctggccct
ggaaggatccctgcagaaacgtggaattgttgaacaatgctgcaccagcatctgttcgttataccagttagagaactactgcaactag
# Scores:  ['sd16sRNADuplexMFE: -8.4', 'utrStructureMFE: -10.4', 'cdsCAI: 0.346776020332']
# Levels:  ['sd16sRNADuplexMFELevel: 1', 'utrStructureMFELevel: 2', 'cdsCAILevel: 3']
# Number of generated solutions:  66
# Distance to seed:  49
############################

Program finished...
```

Figure 6.8 Running TranslationFeatureEcoliDesigner (Optimization design)

Lastly, D-Tailor can also be used to randomly sample the sequence space. The user can indicate a number of sequences to generate given the mutational constraints (Figure 6.9). This option is useful to sample the sequence space.

```
macbook:D-Tailor jcg$ PYTHONPATH=. python
RunningExamples/Designer/TranslationFeaturesEcoliDesigner.py randomsampling 1000
time elapsed: 61.98 (s)    solutions generated: 297          rate (last min.): 4.79 sol/s
rate (overall): 4.79 sol/s
time elapsed: 134.74 (s)   solutions generated: 665          rate (last min.): 5.06 sol/s
rate (overall): 4.94 sol/s
time elapsed: 199.14 (s)   solutions generated: 962          rate (last min.): 4.61 sol/s
rate (overall): 4.83 sol/s
RandomSampling: 1000 solutions generated.
Program finished...
```

Figure 6.9 Running *TranslationFeaturesEcoliDesigner* (RandomSampling design)

6.5. Designer algorithm

The algorithm that generates the desired sequences is implemented by the method *run* in *SequenceDesigner*. Briefly, the algorithm loops through the evolution cycle

step until it finds all the user-defined combinations of features scores. The pseudocode and a schematic of the algorithm are presented in Figure 6.10 and Figure 6.11, respectively.

Each evolution cycle consist of three steps: i) a particular target that is yet to be found is selected (Figure 6.11, step 1); ii) the repository of sequences previously generated (including the seed sequence) is searched to select a template sequence amongst the closest to target in the feature space (Figure 6.11, step 2); iii) a defined number of mutational iterations starting with the selected template sequence is performed (Figure 6.11, step 3).

```
combinations_to_find = all_combinations

while combinations_to_find != []:
     desired_combination = getElement(combinations_to_find)
     #get a sequence already generated that is close to the desired combination in the feature
space.
     solution = getSolutionFromDataBaseCloserTo(desired_combination)

     #Evolution cycle
     while solution != desired_combination or iteration != MAX_ITERATIONS:
          old_solution = solution
          solution = solution.mutate() #mutate current solution to get a new sequence
          DataBase.store(solution)
          If solution.combination in combinations_to_find:
               Combinations_to_find.remove(solution.combination)
          If selection == 'directional':
               If distanceToDesiredCombination(solution) <
distanceToDesiredCombination(old_solution):
                  #use previous sequence if the new sequence is farther away from desired
combination
                  solution = old_solution
          else if selection == 'neutral':
               #randomly select the template sequence for next iteration
               solution = choice(solution,old_solution)
```

Figure 6.10 Pseudocode of *SequenceDesigner* algorithm

In each mutational iteration the current sequence is evaluated to find what feature scores need to be changed to match the desired combination. Then, one of the non-matching features is randomly selected to apply a mutation to the current sequence. Here, two types of mutations can be applied: 1) targeted mutation, and 2) oriented mutation. The former specifically indicates regions of the sequence that are more likely to alter feature score, whereas the latter consists in applying more directed mutations that will move a feature towards the desired level (see section 6.1). Then, the features scores of the resulting sequence are computed using the Analyzer module. Here, if the new sequence matches the target combination then the sequence is validated, the target is marked as completed and the evolution cycle is terminated. Otherwise, one of the two sequences (the template or the mutated one) will be used in the next iteration of the evolution cycle depending on the selection option (below).

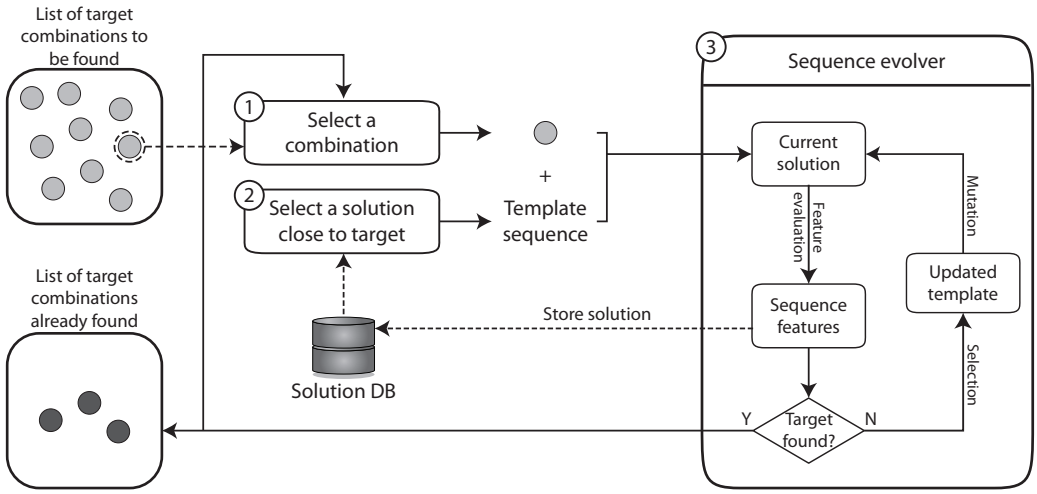


Figure 6.11 Schematic of the *SequenceDesigner* algorithm

We then used D-Tailor to analyze the referred features across the entire *E. coli* genome (Figure 5.4). We set five different levels for each feature based on their respective quintiles (Table 1). A full-factorial design based on such configuration yields a total of 125 (5^3) different target combinations of all feature levels across the three variables. We next randomly selected 30 gene sequences along with their 5'UTR from *E. coli* and compared four different design strategies within D-Tailor to generate a set of sequences conforming to a full-factorial design for each of the seeds. We defined that designed sequences could be generated by unrestricted mutations in the entire 5'UTR region, composed by 49 nucleotides, but only synonymous mutations were allowed in the gene coding sequence.

We first used the most rudimentary design strategy available in D-Tailor, random sampling, to generate random sequences until the 125 different targets were found. Every attempt to complete this design goal using this purely random procedure was aborted after 3,000 generated sequences due to its obvious inefficiency (Figure 6.12A-B, black solid and faded lines). The second design strategy included the canonical heuristic algorithm implemented by D-Tailor (see below) and used the simplest mutational method, wherein new sequences are consecutively generated by random mutation (Figure 6.12B, yellow line). This strategy significantly improved the efficiency of the search algorithm as compared to that of the random sampling method. Nonetheless, the overall performance of the algorithm was still modest since many sequences had to be generated to find the required targets. The third mutational strategy remarkably improved the search algorithm efficiency by employing spatially targeted mutations that more rapidly evolve a sequence towards some desired features scores target (Figure 6.12B, light blue line). Lastly, a fourth

strategy using more 'rational' mutational operators that orient mutations toward the desired target provided slightly faster dynamics (Figure 6.12A-B, orange solid and lines).

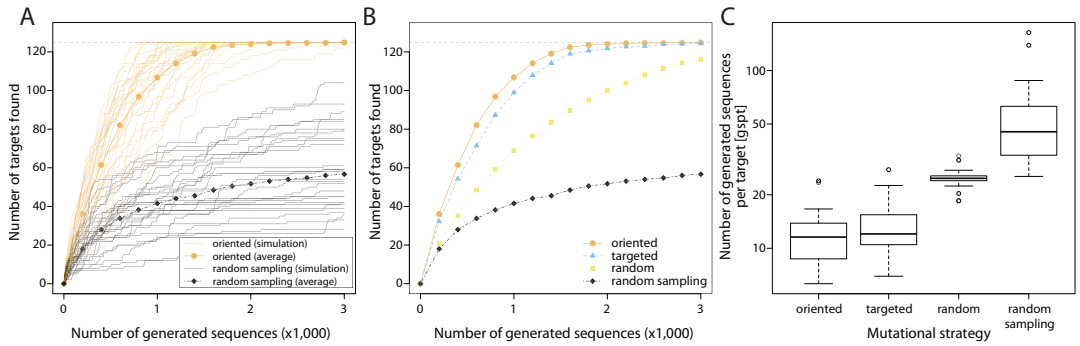


Figure 6.12 Mutational strategies performances

(A) Number of combinations found (out of 125) as a function of the number of generated sequences. Two different mutational strategies are depicted: oriented (orange) and random sampling (black). For each strategy, we performed 30 different simulation of a full-factorial design (faded lines) using different seed sequences. The solid lines represent the average number of target combinations found (across 30 replicates) as the number of generated sequences increases. (B) The average performance of the four different mutational strategies. (C) The number of generated sequences per target combination found using the different mutational strategies.

One other important functionality available to the user is the option to define the selection bias in the heuristic algorithm. There are two different options available:

- *neutral* — the sequence to be used as a template in the next mutational iteration is randomly selected between the template of the current iteration or the newly resulting sequence.
- *directional* — the new sequence is always generated from the previously mutated sequence as long as it is closer to the desired combination.

The selection option can be selected when calling the method *run()* using the parameter *selection*.

We evolved the initial 30 seed sequences toward six different target combinations bearing different distances to the seed in the feature space (Figure 6.13). Then, we examined the behavior of the algorithm in response to the two contrasted selective regimes: neutral and directional selection. As expected, when using D-Tailor with a less constrained selection (i.e. neutral), it was necessary to generate more sequences in order to find the target combination(s). This is because the relaxed selection will necessary generate less diversity and, hence, take more evolution cycles to find the target (Figure 6.13B and D). Naturally, the final designed sequences using the neutral selection option will be more similar to the initial seed sequence (measured

using the hamming distance to the seed sequence) than when using the option directional selection (Figure 6.13A and C).

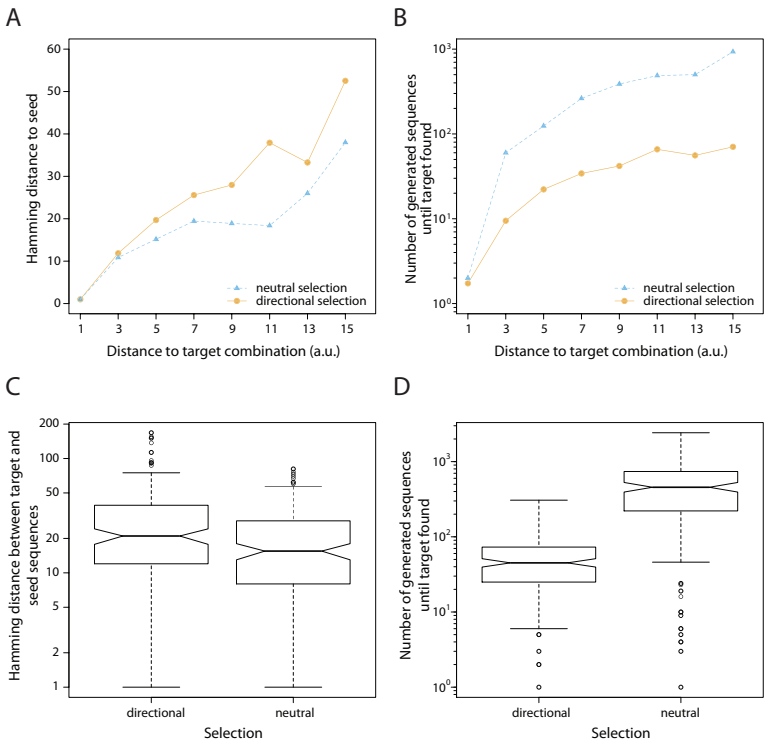


Figure 6.13 Selection options in *SequenceDesigner*

(A) The different lines show the average hamming distance between the seed and the sequence matching the target combination as a function of the distance (in the feature space) to the target combination using neutral (light blue) or directional (orange) selection. (B) The number of generated sequences until the desired target is found as a function of the distance (in the feature space) to the target combination using either neutral (light blue) or directional selection (orange). (C and D) The hamming distance (C) and number of generated sequences until target is found (D) for the 30 different simulations using either directional or neutral selection.

6.6. Designer results

D-Tailor uses an SQLite database to store all the generated sequences and their features. That way generated solutions can be accessed using a standard SQLite client (e.g. SQLite Manager add-on for Firefox) or use some D-Tailor utils to export the generated sequences and retrieve statistics (Figure 6.14). The following utilities are available in the package *Utils*:

- DB2CSV — exports the tables containing all the generated solutions and desired combination to a CSV file specified by the user;
- DB2FASTA — exports all the generated sequences to FASTA format;
- DBStatistics — a script that can be used to query a SQLite instance and print the number of sequences generated and different combinations found;

- DBKinetics — prints a time series of defined size (default 50) with the number of generated solutions and combinations found over time.

```
######
# DB2CSV

macbook:D-Tailor jcg$ PYTHONPATH=. python Utils/DB2CSV.py testFiles/outputFiles/tfec_1.sqlite
Generating CSV files for testFiles/outputFiles/tfec_1.sqlite ... Done!

macbook:D-Tailor jcg$ head -n 3 testFiles/outputFiles/tfec_1.sqlite.generated_solutions.csv
generated_solution_id,des_solution_id,sequence,sd16sRNADuplexMFE,utrStructureMFE,cdsCAI,sd16sRNADu
plexMFELevel,utrStructureMFELevel,cdsCAILevel,sd16sRNADuplexMFEPosition,utrStructureMFEPosition,cd
sCAIPosition,worker_id
100470516384773921758647475759448978081,1.3.2,ttattaccggacaataatatttcaattcattaaagaggagaaaggtaccatg
gccctgtggatgcgcctcctgcccctgctggcgctgctggccctctggggacctgacccagccgcagcctttgtgaaccaacacctgtgcggctcaca
cctggtggaagctctctacctagtgtgcggggaacgaggcttcttctacacacccaagacccgccgggaggcagaggacctgcaggtggggcaggtgg
agctgggcgggggccctggtgcaggcagcctgcagcccttggccctggaggggtccctgcagaagcgtggcattgtggaacaatgctgtaccagcatc
tgctccctctaccagctggagaactactgcaactag,-8.4,-9.9,0.32,1,3,2,0.59,-0.94, 0.72,
249172630681921635831887521585739395265
285501686618022385962284569925274555241,1.2.2,ttattaccggacaataatatttcaattcattaaagaggagaaaggtaccatg
gccctgtggatgcgcctcttacccctgctggcgctgctggccctctggggacctgacccagccgcagcctttgtgaaccaacacctgtgcggctcaca
cctggtggaagctctctacctagtgtgcggggaacgaggcttcttctacacacccaagacccgccgggaggcagaggacctgcaggtggggcaggtgg
agctgggcgggggccctggtgcaggcagcctgcagcccttggccctggaggggtccctgcagaagcgtggcattgtggaacaatgctgtaccagcatc
tgctccctctaccagctggagaactactgcaactag,-8.4,-10.8,0.32,1,2,2,0.59,0.24,0.48,
249172630681921635831887521585739395265

macbook:D-Tailor jcg$ head -n 3 testFiles/outputFiles/tfec_1.sqlite.design_list.csv
des_solution_id,sd16sRNADuplexMFELevel,utrStructureMFELevel,cdsCAILevel,status,worker_id,start_tim
e
1.1.1,1,1,1,DONE,249172630681921635831887521585739395265,None
1.1.3,1,1,3,DONE,249172630681921635831887521585739395265,None

######
# DB2FASTA

macbook:D-Tailor jcg$ PYTHONPATH=. python Utils/DB2FASTA.py testFiles/outputFiles/tfec_1.sqlite
Generating FASTA file(s) for testFiles/outputFiles/tfec_1.sqlite ... Done!

macbook:D-Tailor jcg$ head -n 4 testFiles/outputFiles/tfec_1.sqlite.generated_solutions.fa
>100470516384773921758647475759448978081 | 1.3.2
TTATTACCGGACAATAATATTTCAATTCATTAAAGAGGAGAAAGGTACCATGGCCCTGTGGATGCGCCTCCTGCCCCTGCTGGCGCTGCTGGCCCTCT
GGGGACCTGACCCAGCCGCAGCCTTTGTGAACCAACACCTGTGCGGCTCACACCTGGTGGAAGCTCTCTACCTAGTGTGCGGGGAACGAGGCTTCTTC
TACACACCCAAGACCCGCCGGGAGGCAGAGGACCTGCAGGTGGGGCAGGTGGAGCTGGGCGGGGGCCCTGGTGCAGGCAGCCTGCAGCCCTTGGCCCT
GGAGGGGTCCCTGCAGAAGCGTGGCATTGTGGAACAATGCTGTACCAGCATCTGCTCCCTCTACCAGCTGGAGAACTACTGCAACTAG
>285501686618022385962284569925274555241 | 1.2.2
TTATTACCGGACAATAATATTTCAATTCATTAAAGAGGAGAAAGGTACCATGGCCCTGTGGATGCGCCTCTTACCCCTGCTGGCGCTGCTGGCCCTCT
GGGGACCTGACCCAGCCGCAGCCTTTGTGAACCAACACCTGTGCGGCTCACACCTGGTGGAAGCTCTCTACCTAGTGTGCGGGGAACGAGGCTTCTTC
TACACACCCAAGACCCGCCGGGAGGCAGAGGACCTGCAGGTGGGGCAGGTGGAGCTGGGCGGGGGCCCTGGTGCAGGCAGCCTGCAGCCCTTGGCCCT
GGAGGGGTCCCTGCAGAAGCGTGGCATTGTGGAACAATGCTGTACCAGCATCTGCTCCCTCTACCAGCTGGAGAACTACTGCAACTAG

######
# DBKinetics

macbook:D-Tailor jcg$ PYTHONPATH=. python Utils/DBKinetics.py testFiles/outputFiles/tfec_1.sqlite
Generated Solutions       Desired Solutions Found
0       0
148     14
296     23
444     32
…
7400    124
7418    125

######
# DBStatistics

macbook:D-Tailor jcg$ PYTHONPATH=. python Utils/DBStatistics.py
testFiles/outputFiles/tfec_1.sqlite
testFiles/outputFiles/tfec_1.sqlite       7418    125     0.0168509032084
```

Figure 6.14 D-Tailor utilities

D-Tailor provides one more tool that will be essential for full-factorial designs. Because the number of combinations required by these designs can be extremely hard to achieve, many solutions may be generated during the design process. For example, to generate a full factorial design for the three features affecting translation with 5 levels each (a total of 125 combinations) it was necessary to generate an average of approximately 1500 solutions across 30 different seeds (Figure 6.12). The generation of many sequences will pose an *a posteriori* challenge, which is the selection of only one sequence per combination when one has many to choose from. D-Tailor includes one utility called *ComputeMinimalSet* that precisely addresses this problem. This tool encodes a Monte-Carlo method to select exactly one sequence for each desired combination, such that the total hamming distance between all the sequences is minimized (Figure 6.15).

```
macbook:D-Tailor jcg$ PYTHONPATH=. python Utils/ComputeMinimalSet.py
./testFiles/outputFiles/tfec_2.sqlite.generated_solutions.csv
stop random
1 (1):   550742   -1273
3 (2):   549469   -156
5 (2):   549313   -524
6 (1):   548789   -92
…
################### Summary ###################
number of combinations: 125
average distance nt: 38.21 +/- 20.25
###############################################

macbook:D-Tailor jcg$ ls testFiles/outputFiles/
tfec_2.sqlite
tfec_2.sqlite.design_list.csv
tfec_2.sqlite.generated_solutions.csv
tfec_2.sqlite.generated_solutions.pkl0
tfec_2.sqlite.generated_solutions.pkl1
tfec_2.sqlite.generated_solutions_min_set.fas
tfec_2.sqlite.generated_solutions_min_set_distance_matrix_nt.csv
tfec_2.sqlite.generated_solutions_min_set_feats.csv
```

Figure 6.15 Computing the minimal set

The tool receives a CSV file with all the sequences generated by D-Tailor (obtained using the DB2CSV script) and will generate the following files, where X is the name of the CSV file:

- X.generated_solutions_min_set_feats.csv — a CSV file with the final set of sequences selected;
- X .generated_solutions_min_set.fas — a FASTA file with final set of sequences selected;
- X.generated_solutions_min_set_distance_matrix_nt.csv — a file containing the nucleotide distance matrix between all the selected features.

Appendix II

Protein Abundance Prediction Model: Factors Considered and Correlations

Supplementary Table S1: Factors considered and their correlation with protein abundance

Pearson correlation coefficients between the factors considered and protein abundance (PA) or PA given mRNA levels. F-test p-values were adjusted using false discovery rate (FDR) method to correct for multiple testing.

Category	Variable	Description	Cor. PA	p.value	Partial Cor. PA ~ mRNA	p.value
mRNA	mrna	mRNA transcript abundance	0.7262	0.0000	0.0000	1.0000
TIR	exterior_loop_dg	Binding free energy of the base pair closer to the start codon in the 16S:SD complex	0.1075	0.0022	0.1239	0.0004
TIR	rbs_calc	RBS calculator score as described in (Salis et al., 2009)	0.1462	0.0000	0.1101	0.0017
TIR	accessibility_avg	Average number of single stranded nucleotides in the region [-13,30] with respect to start codon	0.0606	0.0845	0.0912	0.0093
TIR	single	Number of single bases of the structure in the region [-13,30] with respect to start codon	0.0630	0.0729	0.0884	0.0118
TIR	fe	Minimum folding energy (MFE) of the structure in the region [-13,30] with respect to start codon (window with highest correlation with PA)	0.0635	0.0705	0.0729	0.0380
TIR	num_hel	Number of hairpins of the structure in the region [-13,30] with respect to start codon	0.0201	0.5683	0.0458	0.1925
TIR	spacer	Number of bases between the 16S:SD complex and the start codon	-0.0180	0.6090	-0.0371	0.2917
TIR	nOfStacks	Number of stacks in the helix formed by the 16S:SD complex	0.0467	0.1836	0.0366	0.2980
TIR	bulge_mrna	Number of stacks in the helix formed by the 16S:SD complex (mRNA strand)	-0.0810	0.0211	-0.0352	0.3170
TIR	nOfBulges	Total number of bulges in the helix formed by the 16S:SD complex	-0.0733	0.0369	-0.0318	0.3661
TIR	bulge_16s	Number of stacks in the helix formed by the 16S:SD complex (16S strand)	-0.0640	0.0684	-0.0253	0.4711
TIR	sd_spacing	Number of bases between SD motif and the start codon	0.0051	0.8838	-0.0087	0.8039
TIR	sd_score	Sequence score of SD sequence based on the E. coli SD position weight matrix (PWM) in (Shultzaberger et al., 2001)	0.0315	0.3707	-0.0009	0.9793
TIR	hyb_en	MFE of the helix formed between 16S rRNA and the Shine-Dalgarno (SD) sequence	-0.0218	0.5361	0.0008	0.9816
CDS	cai	Codon Adaptation Index (CAI)	0.5828	0.0000	0.3019	0.0000
CDS	ATC	Percentage of occurrences of codon: ATC	0.3974	0.0000	0.2608	0.0000
CDS	GAA	Percentage of occurrences of codon: GAA	0.3215	0.0000	0.2472	0.0000
CDS	Ile	Percentage of occurrences of amino acid: Ile	0.1933	0.0000	0.2316	0.0000
CDS	Glu	Percentage of occurrences of amino acid: Glu	0.2940	0.0000	0.2208	0.0000
CDS	ACG	Percentage of occurrences of codon: ACG	-0.3069	0.0000	-0.2143	0.0000
CDS	CGG	Percentage of occurrences of codon: CGG	-0.3498	0.0000	-0.2092	0.0000
CDS	cai_ramp	CAI of the first 33 codons (ramp)	0.3613	0.0000	0.2083	0.0000
CDS	cu	Codon Usage Bias	0.3704	0.0000	0.2075	0.0000
CDS	AGT	Percentage of occurrences of codon: AGT	-0.2490	0.0000	-0.2016	0.0000
CDS	GGT	Percentage of occurrences of codon: GGT	0.3670	0.0000	0.1728	0.0000
CDS	GGG	Percentage of occurrences of codon: GGG	-0.3181	0.0000	-0.1719	0.0000
CDS	TTG	Percentage of occurrences of codon: TTG	-0.3047	0.0000	-0.1713	0.0000

Category	Variable	Description	Cor. PA	p.value	Partial Cor. PA ~ mRNA	p.value
CDS	CAC	Percentage of occurrences of codon: CAC	0.1818	0.0000	0.1703	0.0000
CDS	GGA	Percentage of occurrences of codon: GGA	-0.3048	0.0000	-0.1697	0.0000
CDS	ACT	Percentage of occurrences of codon: ACT	0.2810	0.0000	0.1655	0.0000
CDS	a_content_init	A content in the region [7,85] (region with highest correlation with PA)	0.2122	0.0000	0.1627	0.0000
CDS	AAT	Percentage of occurrences of codon: AAT	-0.2525	0.0000	-0.1591	0.0000
CDS	TCG	Percentage of occurrences of codon: TCG	-0.2794	0.0000	-0.1564	0.0000
CDS	TTC	Percentage of occurrences of codon: TTC	0.2907	0.0000	0.1471	0.0000
CDS	CGA	Percentage of occurrences of codon: CGA	-0.2870	0.0000	-0.1442	0.0000
CDS	GAT	Percentage of occurrences of codon: GAT	-0.1376	0.0001	-0.1425	0.0000
CDS	CGT	Percentage of occurrences of codon: CGT	0.2378	0.0000	0.1400	0.0001
CDS	Gln	Percentage of occurrences of amino acid: Gln	-0.1869	0.0000	-0.1389	0.0001
CDS	GAC	Percentage of occurrences of codon: GAC	0.2740	0.0000	0.1326	0.0002
CDS	TCT	Percentage of occurrences of codon: TCT	0.3027	0.0000	0.1288	0.0002
CDS	AGG	Percentage of occurrences of codon: AGG	-0.1618	0.0000	-0.1275	0.0003
CDS	Ser	Percentage of occurrences of amino acid: Ser	-0.1503	0.0000	-0.1263	0.0003
CDS	TCA	Percentage of occurrences of codon: TCA	-0.2069	0.0000	-0.1248	0.0004
CDS	CTG	Percentage of occurrences of codon: CTG	0.1657	0.0000	0.1242	0.0004
CDS	TGG	Percentage of occurrences of codon: TGG	-0.2292	0.0000	-0.1220	0.0005
CDS	Trp	Percentage of occurrences of amino acid: Trp	-0.2292	0.0000	-0.1220	0.0005
CDS	CCT	Percentage of occurrences of codon: CCT	-0.1541	0.0000	-0.1205	0.0006
CDS	GTC	Percentage of occurrences of codon: GTC	-0.1490	0.0000	-0.1193	0.0007
CDS	TAT	Percentage of occurrences of codon: TAT	-0.2166	0.0000	-0.1190	0.0007
CDS	CAA	Percentage of occurrences of codon: CAA	-0.2629	0.0000	-0.1164	0.0009
CDS	stop.TAG	Identity of stop codon: TAG	-0.1444	0.0000	-0.1141	0.0011
CDS	AGA	Percentage of occurrences of codon: AGA	-0.1700	0.0000	-0.1141	0.0011
CDS	CCC	Percentage of occurrences of codon: CCC	-0.2679	0.0000	-0.1098	0.0017
CDS	ATA	Percentage of occurrences of codon: ATA	-0.2412	0.0000	-0.1079	0.0021
CDS	GCT	Percentage of occurrences of codon: GCT	0.2708	0.0000	0.1078	0.0021
CDS	GGC	Percentage of occurrences of codon: GGC	0.1283	0.0002	0.1076	0.0022
CDS	GTT	Percentage of occurrences of codon: GTT	0.3416	0.0000	0.1043	0.0029
CDS	CAG	Percentage of occurrences of codon: CAG	-0.0534	0.1289	-0.0968	0.0058
CDS	ACA	Percentage of occurrences of codon: ACA	-0.1877	0.0000	-0.0944	0.0072
CDS	TCC	Percentage of occurrences of codon: TCC	0.1866	0.0000	0.0938	0.0075
CDS	AAG	Percentage of occurrences of codon: AAG	0.0350	0.3190	-0.0903	0.0101
CDS	AGC	Percentage of occurrences of codon: AGC	-0.1822	0.0000	-0.0889	0.0113
CDS	TGT	Percentage of occurrences of codon: TGT	-0.1621	0.0000	-0.0879	0.0122
CDS	ATT	Percentage of occurrences of codon: ATT	-0.0737	0.0360	0.0816	0.0201
CDS	GCC	Percentage of occurrences of codon: GCC	-0.2478	0.0000	-0.0815	0.0202
CDS	TTT	Percentage of occurrences of codon: TTT	-0.2115	0.0000	-0.0813	0.0206
CDS	AAC	Percentage of occurrences of codon: AAC	0.2047	0.0000	0.0792	0.0241
CDS	TTA	Percentage of occurrences of codon: TTA	-0.2908	0.0000	-0.0769	0.0285
CDS	CCG	Percentage of occurrences of codon: CCG	0.0638	0.0694	0.0756	0.0313
CDS	stop.TAA	Identity of stop codon: TAA	0.1513	0.0000	0.0752	0.0323

Category	Variable	Description	Cor. PA	p.value	Partial Cor. PA ~ mRNA	p.value
CDS	CTT	Percentage of occurrences of codon: CTT	-0.1981	0.0000	-0.0746	0.0336
CDS	Gly	Percentage of occurrences of amino acid: Gly	0.1233	0.0004	0.0737	0.0359
CDS	CCA	Percentage of occurrences of codon: CCA	-0.1467	0.0000	-0.0728	0.0383
CDS	CTA	Percentage of occurrences of codon: CTA	-0.2366	0.0000	-0.0697	0.0474
CDS	GCG	Percentage of occurrences of codon: GCG	-0.1320	0.0002	-0.0680	0.0528
CDS	His	Percentage of occurrences of amino acid: His	-0.0280	0.4264	0.0672	0.0558
CDS	Thr	Percentage of occurrences of amino acid: Thr	-0.0633	0.0717	-0.0669	0.0568
CDS	GTA	Percentage of occurrences of codon: GTA	0.2030	0.0000	0.0665	0.0583
CDS	GTG	Percentage of occurrences of codon: GTG	-0.0835	0.0174	-0.0606	0.0845
CDS	Asn	Percentage of occurrences of amino acid: Asn	-0.0327	0.3526	-0.0600	0.0879
CDS	Pro	Percentage of occurrences of amino acid: Pro	-0.1617	0.0000	-0.0566	0.1072
CDS	CAT	Percentage of occurrences of codon: CAT	-0.1894	0.0000	-0.0542	0.1227
CDS	CTC	Percentage of occurrences of codon: CTC	-0.1789	0.0000	-0.0538	0.1255
CDS	TAC	Percentage of occurrences of codon: TAC	0.1164	0.0009	0.0523	0.1366
CDS	Phe	Percentage of occurrences of amino acid: Phe	0.0637	0.0700	0.0513	0.1442
CDS	Tyr	Percentage of occurrences of amino acid: Tyr	-0.0789	0.0246	-0.0510	0.1469
CDS	start.GTG	Identity of start codon: GTG	-0.0172	0.6248	-0.0499	0.1559
CDS	start.ATG	Identity of start codon: ATG	0.0063	0.8573	0.0490	0.1629
CDS	Leu	Percentage of occurrences of amino acid: Leu	-0.2374	0.0000	-0.0429	0.2225
CDS	TGC	Percentage of occurrences of codon: TGC	-0.0553	0.1154	0.0420	0.2319
CDS	prot_len	Length of protein (in amino acids)	-0.1536	0.0000	0.0350	0.3191
CDS	CGC	Percentage of occurrences of codon: CGC	-0.1041	0.0030	0.0348	0.3216
CDS	GAG	Percentage of occurrences of codon: GAG	0.0552	0.1162	0.0326	0.3536
CDS	Ala	Percentage of occurrences of amino acid: Ala	-0.0448	0.2021	-0.0301	0.3923
CDS	ATG	Percentage of occurrences of codon: ATG	0.0606	0.0844	0.0296	0.3997
CDS	Met	Percentage of occurrences of amino acid: Met	0.0606	0.0844	0.0296	0.3997
CDS	Lys	Percentage of occurrences of amino acid: Lys	0.2190	0.0000	-0.0240	0.4941
CDS	Cys	Percentage of occurrences of amino acid: Cys	-0.1303	0.0002	-0.0211	0.5476
CDS	Asp	Percentage of occurrences of amino acid: Asp	0.0871	0.0131	-0.0169	0.6311
CDS	stop.TGA	Identity of stop codon: TGA	-0.0800	0.0226	-0.0157	0.6561
CDS	AAA	Percentage of occurrences of codon: AAA	0.2432	0.0000	0.0148	0.6743
CDS	ACC	Percentage of occurrences of codon: ACC	0.0563	0.1091	0.0105	0.7652
CDS	GCA	Percentage of occurrences of codon: GCA	0.0418	0.2347	-0.0102	0.7716
CDS	Arg	Percentage of occurrences of amino acid: Arg	-0.1124	0.0013	-0.0100	0.7766
CDS	start.TTG	Identity of start codon: TTG	0.0204	0.5613	-0.0081	0.8189
CDS	at_content	AT content in the region [7,85] (region with highest correlation with PA)	0.0648	0.0651	-0.0008	0.9826
CDS	Val	Percentage of occurrences of amino acid: Val	0.1716	0.0000	-0.0004	0.9899